การประยุกต์ใช้เทคโนโลยีเอเจนท์เคลื่อนที่ในงานประยุกต์แบบกระจาย กรณีศึกษา เอเจนท์ตั๋วโดยสารเครื่องบิน

Air Ticket Agent:

A Case Study of Mobile Agents in Distributed Database System

โดย

นางสาว มีนา รัตนสาครกุล

รหัส 43067054

รายงานนี้เป็นส่วนหนึ่งของโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2545

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การประยุกต์ใช้เทคโนโลยีเอเจนท์เคลื่อนที่ในงานประยุกต์แบบกระจาย กรณีศึกษา เอเจนท์ตั๋วโดยสารเครื่องบิน
นักศึกษา	นางสาวมีนา รัตนสาครกุล
อาจารย์ที่ปรึกษา	ดร. ภัทรชัย ลลิตโรจน์วงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

ระบบการให้บริการขายตั๋วโดยสารเครื่องบินของตัวแทนขายตั๋วในปัจจุบัน เป็นการทำงานกับฐานข้อมูลแบบกระจายของแต่ละสายการบิน ซึ่งผู้ใช้ต้องเชื่อมต่อกับฐานข้อมูลของหลายๆสายการบิน เพื่อทำการค้นคืนข้อมูลราคาตั๋วโดยสารของแต่ละสายการบิน แต่การจะเข้าถึงฐานข้อมูลของแต่ละสายการบินได้ ต้องเป็นไปตามข้อตกลงของแต่ละสายการบิน และในการค้นคืนข้อมูลของแต่ละสายการบิน ก็ต้องทำการเชื่อมต่อกับฐานข้อมูลของแต่ละสายการบินแยกกัน ทำให้ขาดความคล่องตัวในการทำงานกับระบบ

เทคโนโลยีเอเจนท์เคลื่อนที่ เป็นแนวคิดใหม่ที่กำลังได้รับความสนใจอย่างมาก ดังนั้นจึงมีการนำเทคโนโลยีเอเจนท์เคลื่อนที่มาประยุกต์ใช้กับงานหลายๆด้าน เช่นเดียวกับงานด้านการรวบรวมสารสนเทศในโครงสร้างฐานข้อมูลแบบกระจาย ซึ่งช่วยเพิ่มประสิทธิภาพในการทำงานของระบบแบบกระจาย ที่มีการเพิ่มขึ้นของข้อมูลเป็นจำนวนมาก รวมไปถึงงานในการจัดการกับข้อมูลที่เพิ่มขึ้น ในการศึกษาโครงงานนี้จึงได้นำเทคโนโลยีเอเจนท์เคลื่อนที่มาพัฒนางานประยุกต์แบบกระจาย เพื่อช่วยเพิ่มประสิทธิภาพการทำงานของระบบค้นคืนข้อมูลราคาตั๋วโดยสาร โดยทำการพัฒนาเอเจนท์เคลื่อนที่ให้ทำงานในระบบแทนผู้ใช้ เรียกว่า เอเจนท์ตั๋วโดยสาร

Title	Air Ticket Agent : A Case Study of Mobile Agents in Distributed Database System
Student	Miss Meena Rattanasakornkul
Advisor	Dr. Pattarachai Lalitrojwong
Level of study	Master of Science in Information Technology
Major	Information Science
Academic Year	2002

ABSTRACT

When you want to travel by plane, you have to buy an air ticket from a travel agency or directly to the airline agent. The air fare ticket databases belong to each airline so whenever you want the information about air fare, you have to access many air fare ticket databases located in each airline site. We can categorize this scheme as a distributed database system.

In this context, mobile agent technology is considered an enhancement of distributed technologies as it provides powerful and efficient mechanisms to develop applications for distributed and heterogeneous systems. To deal with the increasing amount of data/resources available nowadays, and the large number of tasks which have to be performed to manipulate the data/resources, mobile agent technology offers the possibility of executing these tasks in an automated way with minimal human intervention. This allows the user to concentrate attention on other activities. The solution proposed by mobile agent technology for the distribution problem is approach to apply this technology into an air fare ticket system called the Air Fare Ticketing Agent.

กิตติกรรมประกาศ

โครงการพัฒนาระบบงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยคำแนะนำและคำปรึกษาที่ ดร.ภัทรชัย ลลิตโรจน์วงศ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงการได้ประสิทธิ์ประสาทให้อย่างต่อเนื่อง ผู้พัฒนารู้สึกขอบพระคุณในความกรุณาจากท่านเป็นอย่างมาก และขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

สุดท้ายนี้ ขอขอบคุณเพื่อนๆ นักศึกษาทุกคนที่ได้ให้ความช่วยเหลือและเป็นกำลังใจให้เรื่อยมา

มีนา รัตนสาครกุล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

ในธุรกิจสายการบิน ข้อมูลของแต่ละสายการบินจะถูกจัดเก็บแยกกันโดยอิสระ การดูแลรักษาข้อมูลเป็นหน้าที่ของแต่ละสายการบิน ที่จะดูแลจัดการกับฐานข้อมูลของตนเอง โดยภาพรวมของระบบแล้ว กล่าวได้ว่าฐานข้อมูลของแต่ละสายการบินเป็นฐานข้อมูลเฉพาะที่ ที่มีการจัดเก็บอยู่ในโครงสร้างฐานข้อมูลแบบกระจาย

ดังนั้นเมื่อผู้ใช้ (ผู้โดยสาร หรือ ตัวแทนจำหน่ายตั๋วโดยสาร) ต้องการข้อมูลราคาตั๋วโดยสารเครื่องบิน ผู้ใช้จะต้องทำการติดต่อกับสายการบินทุกสายการบิน ภายใต้ข้อตกลงร่วมกันของแต่ละสายการบิน ทำให้บริษัทที่เป็นตัวแทนจำหน่ายตั๋วโดยสารเครื่องบิน ไม่สามารถให้ข้อมูลราคาตั๋วโดยสารของทุกๆสายการบินที่ผู้โดยสารต้องการได้

ด้วยโครงสร้างฐานข้อมูลแบบกระจายของสายการบินต่างๆนี้เอง ทำให้ต้องมีระบบที่ทำหน้าที่ในการค้นคืนข้อมูลราคาตั๋วโดยสารของแต่ละสายการบิน และทำงานเป็นตัวแทนของผู้ใช้ เพื่อค้นหาตั๋วโดยสารที่ถูกที่สุดได้ ระบบดังกล่าวจำเป็นต้องมีคุณลักษณะที่สำคัญคือ ความสามารถในจัดการกับการกระจายตัวของข้อมูล (Distribution) เข้าถึงการทำงานอย่างอิสระ(Autonomy) ของฐานข้อมูลเฉพาะที่ (Local database)ได้ และสามารถเข้าถึงฐานข้อมูลที่มีความแตกต่างกันในแต่ละที่ (Heterogeneity หมายถึง ความแตกต่างของแบบจำลอง กลไกการเข้าถึงข้อมูล หรือแพลตฟอร์ม)

1.2 วัตถุประสงค์ของการศึกษา

จากโครงสร้างฐานข้อมูลแบบกระจายของแต่ละสายการบิน ระบบที่จะช่วยนำข้อมูลที่ถูกจัดเก็บไว้ในฐานข้อมูลระยะไกลหลายๆที่มาให้แก่ผู้ใช้ เสมือนกับว่าผู้ใช้กำลังทำงานอยู่บนฐานข้อมูลแบบส่วนกลาง โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ถึงความซับซ้อนของการจัดการกับข้อมูลที่กระจายอยู่ในเครือข่ายนั้น ทำให้มีการนำเอาเทคโนโลยีเอเจนท์เคลื่อนที่มาประยุกต์ใช้ในการสร้างงานประยุกต์แบบกระจาย (Distributed application) และประสบความสำเร็จในการนำมาประยุกต์ใช้กับระบบฐานข้อมูลแบบกระจายแล้ว

โครงงานนี้จึงนำเอาแนวคิดการใช้เอเจนท์เคลื่อนที่ในระบบฐานข้อมูลแบบกระจาย มาประยุกต์ใช้เพื่อสร้างเอเจนท์เคลื่อนที่ ให้ทำงานแทนผู้ใช้ในการคืนค้นราคาตั๋วโดยสารเครื่องบินจาก

ฐานข้อมูลของแต่ละสายการบินที่ผู้ใช้ต้องการ และให้ผลลัพธ์เป็นราคาตั๋วเครื่องบินที่ถูกที่สุดตามสายการบินที่ผู้ใช้กำหนด

1.3 ขอบเขตการศึกษา

เพื่อแก้ปัญหาการค้นคืนข้อมูลราคาตั๋วโดยสารเครื่องบินที่อยู่ในฐานข้อมูลแบบกระจายการศึกษาโครงงานนี้จะนำเสนอซอฟต์แวร์เอเจนท์ ที่ทำหน้าที่แทนผู้ใช้ในการค้นคืนข้อมูลราคาตั๋วโดยสารเครื่องบินที่ถูกที่สุด จากฐานข้อมูลของสายการบินต่างๆที่ผู้ใช้ต้องการ

ซอฟต์แวร์เอเจนท์ที่พัฒนาขึ้นนี้ เป็นซอฟต์แวร์เอเจนท์ประเภทเอเจนท์เคลื่อนที่ ทำงานโดยเดินทางจากแม่ข่าย ไปยังฐานข้อมูลระยะไกล เพื่อค้นคืนราคาตั๋วโดยสารในเส้นทางที่ผู้ใช้ต้องการ โดยเอเจนท์เคลื่อนที่จะเก็บรวบรวมข้อมูลราคาตั๋วโดยสารจากฐานข้อมูลของแต่ละสายการบินที่กระจัดกระจายอยู่ในเครือข่ายนี้ และทำงานแบบ Autonomy คือมีความสามารถในการตัดสินใจได้ด้วยตนเอง ณ ขณะที่ทำงานอยู่ในโหนดใดๆ ว่าจะนำข้อมูลนั้นกลับมาหรือไม่ แล้วนำข้อมูลที่รวบรวมมาได้ส่งให้กับเอเจนท์เฉพาะที่ เพื่อนำผลลัพธ์เป็นราคาตั๋วโดยสารที่ถูกที่สุดมานำเสนอให้แก่ผู้ใช้

1.4 ขั้นตอนการดำเนินการศึกษา

1. กำหนดวัตถุประสงค์การศึกษา
2. ศึกษาพื้นฐานการทำงานของเอเจนท์เคลื่อนที่ การประยุกต์ใช้เอเจนท์เคลื่อนที่ในงาน ด้านฐานข้อมูลแบบกระจาย
3. ออกแบบโครงสร้างการทำงานของระบบ เมื่อนำเอเจนท์เคลื่อนที่มาประยุกต์ใช้
4. จำลองระบบฐานข้อมูลแบบกระจาย
5. พัฒนาซอฟท์แวร์เอเจนท์เคลื่อนที่
6. ตรวจสอบผลการทำงานของเอเจนท์เคลื่อนที่ ว่าทำงานได้ถูกต้องตามที่ได้รับมอบหมายหรือไม่
7. สรุปผลการศึกษา

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่ได้รับจากซอฟต์แวร์เอเจนท์ที่พัฒนาขึ้นคือ ผู้ใช้จะได้รับข้อมูลราคาตั๋วโดยสารที่ถูกที่สุด จากการรวบรวมข้อมูลในระบบฐานข้อมูลแบบกระจาย โดยลดความซ้ำซ้อนในการทำงานของผู้ใช้ นอกจากนี้ประโยชน์ที่จะได้รับทางอ้อมจากการประยุกต์ใช้งานเอเจนท์เคลื่อนที่คือ การเพิ่มประสิทธิภาพการประมวลผลของระบบ Multidatabase เพราะว่าเซิร์ฟเวอร์ของเอเจนท์จะทำให้เกิด

Uniform global platform ซึ่งซ่อนความแตกต่างของฐานข้อมูลเฉพาะแต่ละที่ไว้ภายใน ในขณะที่ยังรักษาความสามารถในการควบคุมการทำงานอย่างอิสระของฐานข้อมูลแต่ละที่ไว้ได้

การประยุกต์ใช้เอเจนท์เคลื่อนที่ยังช่วยให้การส่งผ่านข้อมูลในเครือข่ายมีประสิทธิภาพดีขึ้น เพราะข้อมูลที่ส่งผ่านในเครือข่ายจะมีน้อยลง ทำให้ช่องสัญญาณในเครือข่ายมีเพิ่มขึ้น เป็นต้น

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 เอเจนท์เคลื่อนที่

เอเจนท์ เป็นซอฟต์แวร์แบบหนึ่งที่ทำงานหนึ่งๆแทนผู้ใช้หรือเอนทิตีอื่น เพื่อทำการรวบรวม กรอง หรือประมวลผลข้อมูล โดยสามารถทำงานและตอบสนองต่อการเปลี่ยนแปลงของสภาพแวดล้อมได้เองโดยอัตโนมัติ และการทำงานนั้นจะต้องเสร็จสมบูรณ์ (Friedemann and Kotz. 2000) บางครั้งเอเจนท์ก็ต้องอาศัยความร่วมมือจากเอเจนท์หลายๆตัว

คุณลักษณะโดยทั่วไปของเอเจนท์คือ มีจุดมุ่งหมาย (Intention) มีความฉลาด (Intelligence) มีภาวะอิสระ (Autonomy) (คือสามารถทำงานให้สำเร็จได้ด้วยตัวเอง) และสามารถเชื่อถือได้ (Believability) (Dale. 1995)

เนื่องจากเอเจนท์เป็นซอฟต์แวร์ จึงมีข้อจำกัดเช่นเดียวกับที่ซอฟต์แวร์ทั่วไปมี เช่น ความแตกต่างของแพลตฟอร์มการใช้งานร่วมกันของระบบ การเคลื่อนที่ผ่านเครือข่าย เป็นต้น เพื่อแก้ปัญหาดังกล่าว จึงเกิดการนำแนวความคิดของโค้ดเคลื่อนที่ (Mobile code) เข้ามารวมกับเอเจนท์

โค้ดเคลื่อนที่ คือโค้ดที่เดินทางข้ามเครือข่าย และไปกระทำการที่เครื่องอื่น โดยสามารถทำงานอยู่บนแพลตฟอร์มที่หลากหลายได้ โค้ดเคลื่อนที่จะถูกเขียนขึ้นเพียงครั้งเดียวแล้วทำงานแบบอัตโนมัติ ภาษาที่ใช้ในการเขียนโค้ดเคลื่อนที่ ได้แก่ Java, JavaScript, VBScript, ActiveX, Postscript และ Word macros (Edward. 1999)

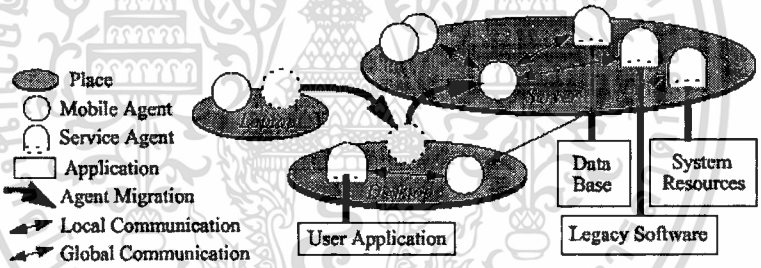
เอเจนท์เคลื่อนที่ (Mobile agent) รวมเอาแนวคิดของเอเจนท์ และโค้ดเคลื่อนที่เข้าด้วยกัน และมีการทำงานในลักษณะการคำนวณแบบกระจาย (Distributed computation)

2.1.1 แนวคิดพื้นฐานของเอเจนท์เคลื่อนที่

เครือข่ายคอมพิวเตอร์ประกอบด้วยโหนดคอมพิวเตอร์มากมายเชื่อมต่อกันผ่านช่องสัญญาณของการสื่อสาร การพัฒนาเครือข่ายคอมพิวเตอร์มีวัตถุประสงค์เพื่อตอบสนองความต้องการในการแลกเปลี่ยนข้อมูลระหว่างโหนดคอมพิวเตอร์และการแบ่งปันทรัพยากรซึ่งกันและกัน แต่ในปัจจุบันผู้ใช้เครือข่ายมีจำนวนมาก ทำให้ทรัพยากรต่างๆไม่เพียงพอ เช่น แบนด์วิดท์ และความเชื่อถือได้ของช่องทางการสื่อสารก็น้อยลง ทำให้เกิดการแลกเปลี่ยนข้อมูลและการแบ่งปันทรัพยากรในเครือข่ายคอมพิวเตอร์รูปแบบใหม่ ซึ่งก็คือ *เอเจนท์เคลื่อนที่*

เอเจนท์เคลื่อนที่ คือโปรแกรมที่สามารถเดินทางข้ามเครือข่ายได้เองโดยอัตโนมัติ และเข้าไปทำงานบางอย่างในคอมพิวเตอร์อื่นที่อยู่ในเครือข่าย ซึ่งต่างจากการทำงานแบบระบบรับ-ให้บริการ (Client/Server) นั่นก็คือ ในการทำงานแบบไคลเอนท์/เซิร์ฟเวอร์ เอนทิตีที่ใช้ในการประมวลผล (เช่น โค้ด) จะถูกติดตั้งอยู่ที่เครื่องๆหนึ่ง ซึ่งแต่ละเครื่องจะมีบทบาทของตนเองชัดเจน คือเซิร์ฟเวอร์จะให้บริการ และมี Client เรียกใช้บริการ การสื่อสารระหว่าง Client และเซิร์ฟเวอร์ใช้วิธีการส่งข้อความ (Message Passing) ซึ่งการทำงานนั้น ผู้เขียนโปรแกรมจะต้องกำหนด Network address ของผู้รับได้ (Dale. 1995) ในขณะที่เอเจนท์เคลื่อนที่จะทำการเคลื่อนย้าย/คัดลอกโปรแกรมและสถานะของเอเจนท์ไปประมวลผลยังเซิร์ฟเวอร์ทำให้การประมวลผลที่เกิดขึ้นอยู่ใกล้กับข้อมูลมาก สารสนเทศที่ได้ก็ไม่ต้องเคลื่อนย้ายกลับมาจากเซิร์ฟเวอร์

2.1.2 ระบบเอเจนท์เคลื่อนที่



รูปที่ 2.1 ระบบเอเจนท์เคลื่อนที่

แบบจำลองของเอเจนท์จะอยู่บนแนวคิดของ "เอเจนท์และสถานที่" (Rothermel and Schwehm. 1998) จากรูปที่ 2.1 จะพบว่า ระบบเอเจนท์ (Agent system) จะอยู่บนสถานที่หนึ่ง (Place) ซึ่งเป็นที่ที่มีบริการต่างๆมากมายแตกต่างกัน ระบบเอเจนท์เคลื่อนที่ประกอบด้วย

(1) เอเจนท์ เอเจนท์เป็นเอนทิตีที่อาจเคลื่อนที่จากสถานที่หนึ่งไปยังสถานที่อื่น เพื่อพบกับเอเจนท์ตัวอื่น หรือเดินทางไปใช้บริการในสถานที่อื่น แบ่งเป็น 2 แบบคือ เอเจนท์เคลื่อนที่ และเอเจนท์บริการ (Service agent) เอเจนท์บริการเป็นเอเจนท์ที่อยู่กับที่ ทำหน้าที่ให้บริการในสถานที่ต่างๆ บริการที่มีแบ่งเป็น 2 ระดับคือ System service จะให้บริการแฟ้มข้อมูล และการเข้าถึงข้อมูลในไดเรกทอรี่ และ Application service เช่น บริการจองห้องพัก บริการสั่งดอกไม้ เป็นต้น ในทางเทคนิคแล้ว เอเจนท์บริการจะแปลงคำขอบริการจากเอเจนท์อื่นให้เป็น "ภาษาเอเจนท์" (Agent language) เพื่อให้บริการเฉพาะบุคคลได้

(2) สถานที่ สถานที่จะให้สภาพแวดล้อมที่ปลอดภัยแก่เอเจนท์ในการประมวลผล โดยทั่วไปแล้ว สถานที่หนึ่งก็คือโหนดหนึ่งของเครือข่ายนั่นเอง แต่ถ้าทุกบริการต่างก็อยู่บนโหนดเดียวกัน โหนดดังกล่าวก็จะจัดว่าเป็น Multiple place ตัวอย่างเช่น มีโหนดอยู่โหนดหนึ่งซึ่งถูกใช้แทนสถานที่หลายๆที่ ซึ่งสถานที่บนโหนดแต่ละแห่งถูกกำหนดให้เป็นของเอเจนท์แต่ละกลุ่ม การเข้าถึงบริการจะต้องใช้ Policy ซึ่งไม่ใช่ว่าทุกความต้องการในสถานที่บนโหนดใดๆ จะต้องมีสถานที่เป็นของตนเอง เมื่อเป็นเช่นนี้ จึงเกิดเอเจนท์ที่ให้บริการสถานที่ โดยให้บริการทั้งแบบเฉพาะที่ (Local) และระยะไกล (Remote)ได้ด้วย

2.1.3 การสื่อสารของเอเจนท์

ความสามารถในการสื่อสารเป็นพื้นฐานของเอเจนท์เคลื่อนที่ ซึ่งมีเกณฑ์ที่ใช้ในการแบ่งหลายแบบด้วยกัน แบบแรกเป็นการแบ่งตามความสามารถในการสื่อสารข้ามเครือข่าย ประกอบด้วย (Dale. 1995)

(1) Network-oriented เอเจนท์จะสื่อสารโดยใช้รูปแบบการสื่อสารผ่านเครือข่าย เช่น การส่งข้อความ หรือ RPC เป็นต้น ซึ่งวิธีนี้ผู้ที่ทำการสื่อสารกันไม่จำเป็นต้องอยู่บนโหนดเดียวกัน หรือบนเครือข่ายเดียวกัน

(2) Node-oriented เอเจนท์จะสื่อสารผ่านรูปแบบการสื่อสารที่เรียกว่า Local interprocess เช่น Files, Shared memory หรือ Anonymous pipes วิธีนี้ผู้ที่ทำการสื่อสารต้องประมวลผลอยู่บนโหนดเดียวกัน

การแบ่งระดับการสื่อสารของเอเจนท์ตามการประสานเวลา (Synchronization) แบ่งเป็น 2 รูปแบบคือ (Dale. 1995)

(1) ประสานเวลา ผู้ที่ทำการสื่อสารกันต้องจัดแบ่งเวลาที่จะใช้ในการสื่อสาร มีการประสานเวลาก่อนที่จะทำการส่งข้อมูล และมีการยืนยันการหลังจากการโอนย้ายข้อมูล วิธีนี้มักใช้เมื่อการโอนย้ายข้อมูลเป็นเรื่องสำคัญ ตัวอย่างเช่น เอเจนท์ที่ต้องการจะขอข้อมูลในฐานความรู้ของผู้อื่น

(2) ไม่ประสานเวลา ผู้ที่จะทำการสื่อสารกันสามารถที่จะติดต่อเมื่อใดก็ได้ตามต้องการ วิธีนี้มักจะใช้กับการโอนย้ายข้อมูลที่เป็นพวกข่าว ผู้รับต้องต้องตรวจสอบข้อมูลที่ได้รับนั้นเอง

2.1.4 การโต้ตอบ

ลักษณะของการโต้ตอบ (Interaction) ของเอเจนท์ในระบบ สามารถแบ่งได้ดังนี้ (Rothermel and Schwehm. 1998)

(1) เอเจนท์กับเอเจนท์บริการ เป็นลักษณะของไคลเอนท์/เซิร์ฟเวอร์ นั่นคือ เอเจนท์บริการจะทำงานตามที่มีการร้องขอจากเอเจนท์อื่น มักจะใช้ RPC เป็นรูปแบบการสื่อสาร

(2) เอเจนท์เคลื่อนที่กับเอเจนท์เคลื่อนที่ การโต้ตอบประเภทนี้ต่างจากแบบแรก เพราะเอเจนท์ทั้งสองจะสื่อสารกันในลักษณะของ Peer to peer มากกว่าที่จะเป็นแบบไคลเอนท์/เซิร์ฟเวอร์ เอเจนท์แต่ละตัวจะมีวิธีควบคุมการโต้ตอบเพื่อบรรลุวัตถุประสงค์ของตัวเองตามต้องการ รูปแบบการสื่อสารที่ใช้ไม่ได้จำกัดอยู่ที่ RPC อย่างเดียว แต่จะขึ้นกับระดับความยืดหยุ่นที่ต้องการมากกว่า จึงอาจจะใช้การส่งข้อความก็ได้

(3) การสื่อสารของกลุ่มเอเจนท์ ในสองแบบแรกเราสมมุติว่าเอเจนท์ที่ทำการโต้ตอบกันนั้นรู้จักกัน นั่นคือ ผู้ส่งข้อความหรือ RPC จะรู้ว่าผู้รับข้อความของตนคือใคร แต่ถ้าผู้ส่งข้อความไม่รู้จักผู้รับ เช่น มีงานอยู่งานหนึ่งที่ให้เอเจนท์กลุ่มหนึ่งเป็นผู้ทำ แต่ละเอเจนท์ในกลุ่มก็รับงานย่อยๆ ไปทำงาน เอเจนท์แต่ละตัวสามารถสร้างกลุ่มของเอเจนท์ย่อยๆขึ้นมาอีก สมมุติว่ามีเอเจนท์บางตัวต้องการจบการทำงานของกลุ่มของตัวเอง เอเจนท์จะต้องส่งคำร้องขอเพื่อขอจบการทำงานออกไปทั้งที่ไม่รู้จักสมาชิกแต่ละตัวในกลุ่มของตัวเอง เพราะเอเจนท์ที่ส่งรู้จักเพียงแต่กลุ่ม แต่ไม่รู้จักตัวเอเจนท์ในกลุ่ม การสื่อสารประเภทนี้จะต้องใช้ Group communication protocol (เช่น ISIS toolkit) ซึ่งผู้ส่งจะส่ง Event message ออกไปโดยที่ไม่รู้จักใคร ส่วนผู้รับก็จะแสดงตัวเองต่อเหตุการณ์ ที่มันสนใจ

(4) ผู้ใช้กับเอเจนท์ เอเจนท์เคลื่อนที่ต้องนำผลลัพธ์ที่ได้กลับมาให้ผู้ใช้ โดยทั่วไปแล้วเอเจนท์เคลื่อนที่จะลงบันทึกข้อมูลไว้กับเอเจนท์ที่ทำงานอยู่บนโหนดก่อน แล้วจึงนำเสนอหรือประมวลผลข้อมูล ซึ่งการรวมข้อมูลและจัดรูปแบบให้ผู้ใช้ มักจะกระทำโดยเอเจนท์ตัวอื่น เอเจนท์ประเภทนี้มักรวมอยู่กับ Desktop environment ของผู้ใช้ เช่น RTF ใน MS Word (Dale. 1995)

เมื่อเอเจนท์ต้องการสื่อสาร จะต้องสร้างความสัมพันธ์ในการสื่อสารที่เรียกว่า “Session” เพื่อใช้ในการประสานเวลากันของเอเจนท์ที่ต้องการจะพบ และทำงานร่วมกัน เราสามารถแบ่ง Session ได้เป็น 2 ประเภท คือ ช่วงเวลาสื่อสารที่ใช้ภายในสถานที่เดียวกัน (Intra-place session) คือ การสื่อสารที่เกิดขึ้นเฉพาะที่เท่านั้น และช่วงเวลาสื่อสารที่ใช้ต่างสถานที่กัน (Interplace session) คือ การสื่อสารครอบคลุม

2.1.5 จุดเด่นของเอเจนท์เคลื่อนที่

เราสามารถนำเอาแนวคิดของเอเจนท์เคลื่อนที่ไปใช้ให้เกิดประโยชน์ เนื่องจากเอเจนท์เคลื่อนที่มีคุณสมบัติที่เป็นข้อดี ดังนี้ (Sahuguet. 1997)

(1) การทำงานแบบไม่ประสานเวลา คือการทำงานที่ทั้งสองฝ่ายไม่จำเป็นต้องเข้าสู่ระบบหรือเครือข่ายในเวลาเดียวกัน เช่น การรับส่งอีเมล ซึ่งเหมาะมากกับผู้ใช้อุปกรณ์ที่เคลื่อนที่ได้ อย่างเช่น พีดีเอ หรือแล็ปท็อปคอมพิวเตอร์ เพราะผู้ใช้มักจะต้องเดินทางตลอดเวลา เมื่อผู้ใช้เชื่อมต่อเข้าไปในเครือข่าย แล้วส่งเอเจนท์ออกไปทำงาน จากนั้นก็ออกจากเครือข่าย เอเจนท์ที่ส่งออกไปนั้นจะเดินทางไปในเครือข่ายเพื่อไปทำงานยังเครื่องระยะไกล โดยการทำงานของเอเจนท์จะประสานเวลากับเครื่องระยะไกล เมื่อเอเจนท์ทำงานเสร็จ อาจจะรอให้แม่ข่าย เรียกมันกลับไป หรืออาจเดินทางกลับด้วยตนเอง

นอกจากนี้ การทำงานแบบประสานเวลายังมีประโยชน์ต่อการสร้างเส้นทางสื่อสารได้ เนื่องจากการสื่อสารที่เป็น Session-based จะบังคับให้ต้องมีการเชื่อมต่อระหว่างผู้ส่งและผู้รับเกิดขึ้นอยู่ตลอดเวลา เอเจนท์จะเก็บประวัติของกระบวนการที่ทำงาน และสร้างการเชื่อมต่อขึ้น แม้ว่าจะไม่มีการทำงานใดๆก็ตาม

(2) การทำงานด้วยภาวะอิสระ หมายถึง การทำงานให้สำเร็จได้ด้วยตัวเอง เอเจนท์เคลื่อนที่สามารถทำงานได้ด้วยตัวเอง โดยไม่ต้องการการเชื่อมต่ออยู่ตลอดเวลา เอเจนท์เคลื่อนที่จะนำเอาโค้ดของตัวเองไป และใช้ความรู้ที่มีตัดสินใจการทำงานเอง ซึ่งความรู้บางส่วนอาจจะได้จาก Master ด้วยก็ได้ ทำให้ช่วยประหยัดเวลาในการเดินทางเป็นอย่างมาก

(3) สถาปัตยกรรมของเอเจนท์เคลื่อนที่ เนื่องจากสถาปัตยกรรมของเอเจนท์เคลื่อนที่เป็นแบบ Decentralized, Distributed และ Light ทำให้การแก้ไขโค้ดของเอเจนท์ซึ่งมีขนาดเล็กเป็นไปได้ง่าย และการทำงานร่วมกันของโปรแกรมขนาดเล็กจะมีประโยชน์ต่อการจัดกำหนดการการทำงานของกระบวนการขนาดใหญ่ ให้สามารถใช้ทรัพยากรได้อย่างคุ้มค่า เพราะสถานีงาน (Workstation) สามารถทำหน้าที่เป็น Agent server ในขณะที่ไม่ได้ทำงานอะไร โดยไม่ได้ก่อให้เกิดค่าใช้จ่ายเพิ่มขึ้น

(4) ความสะดวกในการใช้เครื่องระยะไกล เอเจนท์เคลื่อนที่ได้ประโยชน์จากการทำงานระยะไกล ดังนี้

ซีพียู: การใช้ซีพียูของเครื่องระยะไกล ช่วยให้อุปกรณ์เคลื่อนที่ที่มีซีพียูจำกัด (Thin client) สามารถทำงานได้มากขึ้น

หน่วยความจำ: ในการทำงานที่ต้องใช้หน่วยความจำมาก การเข้าถึงหน่วยความจำของเครื่องระยะไกลจะช่วยแก้ปัญหาได้

ระบบหลายตัวประมวล: ถ้าเครื่องระยะไกลนั้นมีหน่วยประมวลผลหลายตัว เอเจนท์ก็จะสามารถทำการประมวลผลแบบขนานได้

แบนด์วิดท์: สามารถส่งเอเจนท์ไปทำงานในเครือข่ายที่มีช่องสัญญาณกว้างกว่า

ทรัพยากรอื่นๆ: เอเจนท์อื่นสามารถใช้ทรัพยากรเฉพาะที่ได้โดยการควบคุมระยะไกล แต่ต้องระวังเรื่องความปลอดภัยของข้อมูลด้วย

แม้ว่าในทางทฤษฎี เอเจนท์จะมีข้อดีอย่างมากในเรื่องของการแบ่งปันการใช้ทรัพยากรให้คุ้มค่า แต่ก็ต้องแลกกับค่าใช้จ่ายในเรื่องของความปลอดภัยในการดูแลข้อมูลจากเอเจนท์ด้วยเช่นกัน

2.1.6 การประยุกต์ใช้งานเอเจนท์เคลื่อนที่

ก่อนที่จะออกแบบระบบ หรือนำรูปแบบของเอเจนท์เคลื่อนที่ไปใช้ ควรจะทำความเข้าใจในเรื่องของสถาปัตยกรรมและความต้องการของเอเจนท์เคลื่อนที่ ซึ่งมีรายละเอียดดังต่อไปนี้

(1) สถาปัตยกรรม สถาปัตยกรรมโดยรวมของเอเจนท์เคลื่อนที่ขึ้นอยู่กับเครื่องเซิร์ฟเวอร์ที่ติดต่อด้วยนั้นเซิร์ฟเวอร์ควรจะให้สภาพแวดล้อมในการทำงานที่เหมาะสมแก่เอเจนท์ เพื่อให้เอเจนท์สามารถใช้ทรัพยากรที่นี่ได้

เอเจนท์จะถูกปล่อยออกมาทำงานโดยเซิร์ฟเวอร์หรือโปรแกรมประยุกต์ และเดินทางผ่านเครือข่ายด้วยโพรโทคอลหนึ่งๆ เช่น IBM ใช้ Aglet Transfer Protocol (ATP), General Magic ใช้โพรโทคอลของตัวเอง, OMG ใช้โพรโทคอลมาตรฐาน

ขณะที่เดินทาง เอเจนท์จะไม่สามารถทำอะไรได้ แต่เมื่อเดินทางถึงเซิร์ฟเวอร์แล้ว เอเจนท์จะถูก Activate ให้กระทำการ (Execute) เหมือนกับเป็น Local program ทั้งนี้ต้องมีข้อกำหนดในเรื่องของการรักษาความปลอดภัยด้วย ที่จุดนัดพบของเอเจนท์ (Agent meeting point – AMP) เซิร์ฟเวอร์จะให้ทุกอย่างที่เอเจนท์ต้องการ โดยทั่วไปจะประกอบด้วย ซีพียู หน่วยความจำ หน่วยเก็บ และข้อมูลเฉพาะที่ (Local information)

(2) สิ่งที่เอเจนท์เคลื่อนที่ต้องการ การนำเอเจนท์เคลื่อนที่ไปใช้ จำเป็นต้องมีสิ่งต่างๆต่อไปนี้ (Sahuguet. 1997)

ภาษาที่ใช้ได้หลายระบบ (Portable language): โค้ดของเอเจนท์เคลื่อนที่ต้องสามารถทำงานได้ทุกที่โดยไม่ต้องแปลใหม่ เนื่องจากจาวาเป็นภาษาที่สร้างขึ้นเพื่อทำงานบนแพลตฟอร์มที่หลากหลายได้ เพราะมีเครื่องเสมือน (Virtual machine) และยิ่งกว่านั้น รหัสไบต์ (Byte-code) ของจาวาก็ทำให้ภาษาโปรแกรมอื่นๆสามารถใช้จาวาในการพัฒนาได้ง่าย ดังนั้น จึงมีการนำจาวามาใช้ในการพัฒนาเอเจนท์เคลื่อนที่

Interpreted language: ตัวแปลคำสั่งจะทำหน้าที่เป็นเครื่องเสมือน และในการแปลคำสั่ง server สามารถตรวจสอบโค้ดก่อนที่จะกระทำการได้ด้วย

การตรวจสอบ (Introspection): โปรแกรมควรมีความสามารถในการตรวจสอบอ็อบเจกต์ได้ โดยที่อ็อบเจกต์จะมีอินเตอร์เฟซให้โปรแกรมทำการตรวจสอบตัวมันด้วยเช่นกัน สิ่งนี้จะช่วยให้เซิร์ฟเวอร์รู้ว่าอ็อบเจกต์นั้นกำลังจะใช้ทรัพยากรอะไร หรือทำให้อ็อบเจกต์รู้ขีดความสามารถของเซิร์ฟเวอร์เช่น การใช้ Java bean เพื่อเขียนส่วนโปรแกรมหรือคอมโพเนนท์ที่นำกลับมาใช้ใหม่ได้

Late binding: เพราะว่าเอเจนท์เดินทางโดยไม่รู้ว่าเซิร์ฟเวอร์ที่จะเข้าถึงนั้นเป็นเซิร์ฟเวอร์ประเภทใด เอเจนท์เคลื่อนที่จึงจำเป็นต้องใช้ Late binding ในการรวมข้อมูลของอ็อบเจกต์ขณะทำการประมวลผล

การแทนความรู้ร่วม (Common knowledge representation): เพราะว่าเอเจนท์ทำงานด้วยภาวะอิสระให้สำเร็จได้ด้วยตัวเอง โดยนำเอาโค้ดและข้อมูลไปด้วย ดังนั้น จึงต้องมีมาตรฐานในการนำเสนอข้อมูลและความรู้ โดยเฉพาะอย่างยิ่ง ความต้องการให้เอเจนท์ทำงานร่วมกันได้ มาตรฐานที่มีการใช้อยู่คือ Darpa Knowledge Effort, Knowledge Information Format (KIF), Ontolingua, Knowledge and Query Manipulation Language (KQML) และ Electronics Data Interchange (EDI) ข้อดีของการมีข้อตกลงที่แน่นอนในการใช้ภาษาใดภาษาหนึ่ง จะช่วยลดความจำเป็นในการใช้เครื่องแปลภาษา (Translator) ได้

การกำหนดสาระสำคัญเกี่ยวกับสภาพแวดล้อมระยะไกล: นอกจากผู้พัฒนาจะเขียนโปรแกรมเอเจนท์ในส่วนของพฤติกรรมแล้ว ผู้พัฒนาควรจะนำเสนอข้อมูลทรัพยากรที่เอเจนท์ต้องการใช้ที่เครื่องระยะไกลด้วย เนื่องจากการที่จะใช้งานบนเครื่องระยะไกลนั้น เครื่องนั้นควรมีทรัพยากรต่างๆที่เอเจนท์ต้องการ

Adaptive program: การทำงานของเอเจนท์มักถูกจำกัดอยู่กับสภาพแวดล้อมแบบใดแบบหนึ่ง เนื่องจาก ณ ที่นั้นๆ อาจมีทรัพยากรไม่เพียงพอ วิธีแก้ที่ง่ายที่สุดก็คือ การเคลื่อนย้ายเอเจนท์ไปยังที่อื่น แต่อย่างไรก็ตาม เอเจนท์ควรมีความสามารถในการจัดการกับข้อจำกัดเหล่านั้นได้ด้วย

การวางแผนเชิงกลยุทธ์: ในการวางแผนสอบถามฐานข้อมูล ผู้ใช้ควรจะสามารถเลือกทางเลือกที่ดีที่สุดก่อนที่จะส่งเอเจนท์ออกไปทำงาน ในบางกรณีการใช้โค้ดเคลื่อนที่ก็ไม่ใช่วิธีแก้ปัญหาที่ดีที่สุด ผู้ใช้ควรพิจารณาในเรื่องของข้อมูลของงานที่จะทำและโครงแบบของเครือข่ายด้วย

โครงสร้างข้อมูลร่วม: แนวคิดของการทำงานของเอเจนท์คือ การใช้เซิร์ฟเวอร์ที่มีทรัพยากรเตรียมไว้ให้ร่วมกัน เพราะฉะนั้น ผู้ใช้จึงต้องแบ่งปันกันใช้ทรัพยากรอย่างดีที่สุด การล็อคเป็นวิธีหนึ่งที่ดี แต่บางครั้งก็มีข้อจำกัดอยู่มาก

ภาวะพร้อมกัน (Concurrency control): จากความต้องการในการใช้ข้อมูลร่วมกัน ทำให้ต้องควบคุมการเข้าถึงข้อมูลพร้อมๆกัน จึงควรพิจารณาถึงการจัดการเวลาในการเข้าถึงทรัพยากรของเอเจนท์ให้ใช้ได้อย่างคุ้มค่าที่สุด

2.1.7 งานประยุกต์ที่เหมาะในการนำเอเจนท์เคลื่อนที่ไปใช้

การพัฒนาโปรแกรมโดยใช้หลักการของเอเจนท์เคลื่อนที่นั้น เหมาะกับงานหลายประเภท เช่น (Nikhil. 1997)

(1) การรวบรวมข้อมูลแบบกระจาย (Distributed data collection) โดยทั่วไปแล้ว การค้นหาข้อมูลในอินเทอร์เน็ตจะใช้ขั้นตอนวิธีการค้นหาแบบกระจาย (Distributed search algorithm) ซึ่งเราสามารถใช้แนวคิดของเอเจนท์ และเอเจนท์ย่อยๆมาทำงานในลักษณะการประมวลผลแบบขนานได้ โดยที่เอเจนท์หลักจะปล่อยเอเจนท์ย่อยออกมาให้ทำงานแต่ละอย่างเอง แล้วเอเจนท์หลักก็จะทำการประมวลผลลัพธ์ที่เอเจนท์ย่อยๆส่งกลับมา เพื่อสร้างกลุ่มของผลลัพธ์ตามที่ผู้ใช้เลือก

(2) การเฝ้าสังเกต (Monitoring) เราสามารถใช้เอเจนท์ในการตรวจสอบความเปลี่ยนแปลงของทรัพยากรในเครือข่าย หรือทำการรวบรวมข้อมูลต่างๆที่ทรัพยากรเหล่านั้นสร้างขึ้นมา โดยสร้างโปรแกรมให้เอเจนท์เคลื่อนที่ทำงานเป็นช่วงเวลา แล้วนำข้อมูลที่ตรวจสอบได้ในแต่ละโหนดกลับมายังระบบเอเจนท์ ตัวอย่างเช่น การใช้เอเจนท์ตรวจสอบเซิร์ฟเวอร์ที่ให้บริการข่าว เพื่อดูว่าข่าวใดที่ผู้ใช้สนใจ หรืออาจเป็นการตรวจสอบการใช้งานของเครือข่าย จะเห็นได้ว่า ลักษณะเด่นของการทำงานของเอเจนท์ในงานประเภทนี้ คือการทำงานด้วยภาวะอิสระ

(3) การบริการการส่งสารสนเทศในเว็บ (Information delivery) Push technology กำลังเติบโตอย่างรวดเร็วในโลกของเวิลด์ไวด์เว็บ ซึ่งเป็นความสามารถในการให้บริการสารสนเทศที่มีการร้องขอสิ่งที่ต้องการออกไป บริการการให้ข้อมูลหรือสารสนเทศในเวิลด์ไวด์เว็บ เป็นงานหนึ่งที่เหมาะกับการนำเอเจนท์มาใช้ ตัวอย่างของเอเจนท์ที่ใช้ในการทำ Pushing สารสนเทศ คือ Message agent และ Entertainment channel agent ซึ่งเป็น Multimedia-based โดยจะเผยแพร่หรือกระจายข้อมูลให้กับสมาชิกในกลุ่ม

(4) การเจรจา (Negotiation) เอเจนท์สามารถดึงข้อมูลและตัดสินใจได้โดยไม่ได้อาศัยแต่เพียงข้อมูลทางสถิติอย่างเดียว (เช่น ฐานข้อมูล) แต่สามารถดึงข้อมูลจากเอเจนท์อื่นเพื่อประกอบการตัดสินใจได้ นั่นคือ เรากำลังประยุกต์ใช้งานกลุ่มเอเจนท์ ซึ่งเอเจนท์แต่ละตัวจะทำหน้าที่แต่ละอย่างแทนคน โดยมีความรู้และตารางการทำงานของตัวเอง จึงสามารถพูดคุยและตกลงเวลาในการพบกันระหว่างเอเจนท์ได้

(5) การทำรายการ (Transaction) เอเจนท์สามารถทำให้กระบวนการที่มีการทำงานยาวๆ ทำงานได้ง่ายขึ้น ตัวอย่างเช่น ในการทำรายการเตรียมการท่องเที่ยว นอกจากเอเจนท์จะสามารถให้ข้อมูลวัน ราคา และข้อกำหนดอื่นๆในการท่องเที่ยวแก่ผู้ใช้แล้ว ยังสามารถที่จะออกเดินทางไปค้นหาข้อมูลในฐานข้อมูลของสายการบินและโรงแรม เพื่อทำการจองบัตรโดยสารและที่พักให้ งานประยุกต์ในลักษณะนี้ เอเจนท์จะใช้ความสามารถในการทำงานด้วยภาวะอิสระ ซึ่งก็คือการทำงานหนึ่งๆให้สำเร็จได้ด้วยตัวเอง และความสามารถในการทำงานเมื่อผู้ใช้ไม่ได้เชื่อมต่ออยู่ในเครือข่าย

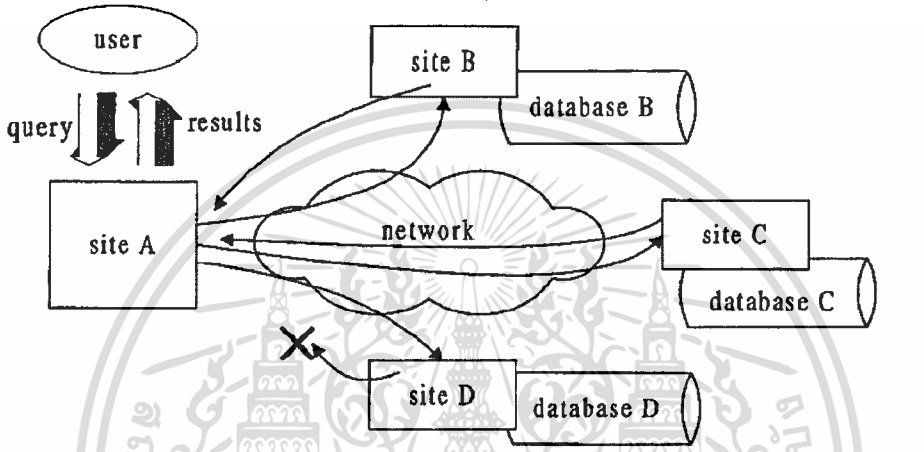
(6) การประมวลผลแบบขนาน (Parallel processing) เอเจนท์ยอมให้มีการแบ่งงานของโปรแกรมประยุกต์เป็นส่วนย่อยๆ แล้วส่งไปให้เอเจนท์ย่อยๆประมวลผลที่เครื่องอื่นที่อยู่ในเครือข่าย (หรืออาจจะเป็นเครื่องของตนเองก็ได้) เมื่องานเหล่านั้นสำเร็จ เอเจนท์ย่อยๆจะมารวมกันเป็น Multi-thread application

2.2 เอเจนท์เคลื่อนที่กับระบบฐานข้อมูลแบบกระจาย (Demartini *et al.* 1999)

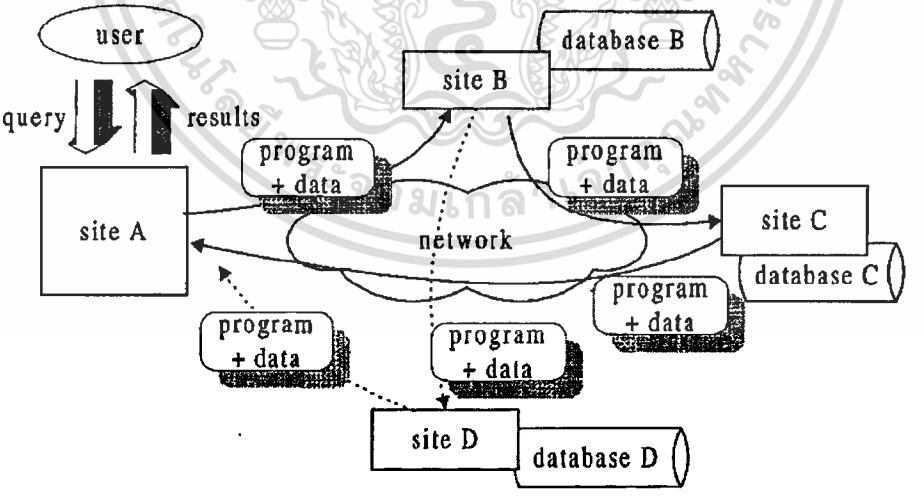
พิจารณาโครงสร้างระบบฐานข้อมูลแบบกระจาย เมื่อฐานข้อมูลมีการกระจายตัวดังภาพที่ 2.2 และการเชื่อมต่อของฐานข้อมูลเป็นแบบ Dial-up network on-demand ซึ่งเชื่อมต่อกันด้วย ISDN เมื่อผู้ใช้สร้างข้อคำถามที่ Site A (สมมติว่ามีการสร้างการเชื่อมต่อของเครือข่ายแล้ว) เพื่อต้องการข้อมูลจาก Site B, C และ D ข้อคำถามที่สร้างขึ้นมาจะถูกส่งไปยังแต่ล่ะ Site เมื่อผลลัพธ์ที่ได้จากแต่ละ Site ถูกส่งกลับมาที่ Site A ก็จะอาจจะมีการนำข้อมูลที่ได้ไปประมวลผลต่อ แต่ถ้ามีบาง Site ไม่ส่งผลลัพธ์กลับมาให้ภายในเวลาที่กำหนด ทางเลือกสำหรับ Site A มีสองทางด้วยกันคือ ถือว่าการส่งข้อคำถามนั้นล้มเหลวแล้วส่งใหม่ทั้งหมด อีกทางเลือกหนึ่งคือนำผลลัพธ์ที่ได้จากบาง Site มาเก็บไว้ก่อน แล้วรอให้การเชื่อมโยงของเครือข่ายใช้ได้เป็นปรกติ จึงส่งข้อคำถามไปยัง Site ที่ขาดการติดต่อใหม่ แล้วจึงนำผลลัพธ์มาประมวลผลต่อไป นอกจากปัญหาที่ผู้ใช้จะต้องรอผลลัพธ์เป็นเวลานานแล้ว การเชื่อมต่อของเครือข่ายแบบ Dial-up network on-demand ที่มี Site อยู่หลาย Site ทำให้การจะเข้าถึงฐานข้อมูลในแต่ละ Site พร้อมๆกัน จำเป็นต้องมีเบอร์โทรศัพท์เท่ากับจำนวน Site ที่จะทำงานด้วย จากปัญหาข้างต้นเมื่อนำเทคโนโลยีเอเจนท์เคลื่อนที่เข้ามาประยุกต์ใช้กับการเข้าถึงข้อมูลในฐานข้อมูลแบบกระจาย โดยที่กำหนดให้การเข้าถึงฐานข้อมูลในเครือข่ายเป็นแบบวนกลับ ดังภาพที่ 2.3 เมื่อผู้ใช้ใน Site A สร้างข้อคำถามส่งไปยัง Site B และเมื่อได้ผลลัพธ์จาก Site B ผลลัพธ์ที่ได้จาก Site B ทำให้รู้ว่าจะต้องเดินทางไปยัง Site C และ D แล้วทำการค้นคืนข้อมูลที่ต้องการต่อ โดยที่ผลลัพธ์ที่ได้จาก Site B จะถูกนำไปด้วย เมื่อเดินทางไปยัง Site ต่างๆที่กำหนดครบแล้ว ก็จะกลับไปยัง Site A พร้อมกับนำผลลัพธ์ที่ได้กลับไป จะพบว่าการเข้าถึงข้อมูลในฐานข้อมูลแบบกระจายโดยใช้เทคโนโลยีเอเจนท์เคลื่อนที่มีข้อดีดังต่อไปนี้

(1) เพิ่มความยืดหยุ่นให้แก่ ข้อคำถาม

(2) ประหยัดค่าใช้จ่ายในการสื่อสาร โดยเฉพาะการเชื่อมต่อแบบ Dial-up เพราะการเชื่อมต่อจะเกิดขึ้นเมื่อเอเจนท์จะเคลื่อนที่เท่านั้น กล่าวคือ เทคโนโลยีเอเจนท์เคลื่อนที่ มีกลไกการที่จะจัดการกับข้อมูลที่มีในเครือข่าย และกลไกการใช้ทรัพยากรที่มีอยู่ในเครือข่าย



รูปที่ 2.2 การเข้าถึงฐานข้อมูลที่กระจัดกระจาย



รูปที่ 2.3 การเดินทางของเอเจนท์เคลื่อนที่แบบไดนามิค

(3) เอเจนท์เคลื่อนที่สามารถทำให้การประมวลผลเกิดขึ้นได้แบบอัตโนมัติ โดยใช้มนุษย์น้อยลง

(4) เอเจนท์เคลื่อนที่สามารถทำงานได้แบบระยะไกล ทำให้สามารถเข้าถึงในฐานข้อมูลที่อยู่ระยะไกลได้แบบ Location transparency เช่นเดียวกับสถาปัตยกรรมไคลเอนท์/เซิร์ฟเวอร์ แต่ความแตกต่างกันอยู่ที่การประมวลผล ซึ่งเอเจนท์เคลื่อนที่จะเคลื่อนย้ายตัวเองไปทำงานยังที่ที่มีข้อมูลเก็บอยู่ แล้วจึงประมวลผลคำขอของผู้ใช้ ทำให้ประสิทธิภาพการทำงานมีมากกว่า

(5) เอเจนท์สามารถที่จะตัดสินใจได้ด้วยตัวเองว่าจะนำข้อมูล ณ ฐานข้อมูลระยะไกลนั้นกลับมาหรือไม่ เป็นการทำงานแบบอิสระ (Autonomous) ซึ่งผู้ใช้สามารถทำงานอย่างอื่นได้ในเวลาเดียวกัน นอกจากนี้เวลาที่จะใช้ในการทำงานให้สมบูรณ์ก็ลดลง เพราะไม่ต้องมีการโต้ตอบระหว่างเอเจนท์เคลื่อนที่กับผู้ใช้

(6) ถ้าหากแม่ข่ายไม่ได้ทำงาน หรือเชื่อมต่อในเครือข่ายชั่วคราว เอเจนท์ก็ยังทำงานต่อได้ เรียกว่า การทำรายการแบบไม่ประสานเวลา (Asynchronous Transaction)

ตัวอย่างข้างต้นแสดงให้เห็นว่าเทคโนโลยีเอเจนท์เคลื่อนที่สามารถแก้ไขปัญหาการจัดการกับฐานข้อมูลที่กระจัดกระจายได้มาก ดังนั้นในโครงงานนี้จึงนำเอาแนวคิดดังกล่าวมาพัฒนางานประยุกต์ที่เป็นเอเจนท์เคลื่อนที่ ทำหน้าที่ช่วยผู้ใช้ในการเข้าถึงข้อมูลในฐานข้อมูลที่กระจัดกระจาย

2.3 Aglet Workbench (Danny and Mitsuru. 1998)

เครื่องมือที่ใช้ในการพัฒนาเอเจนท์เคลื่อนที่มีอยู่มากมาย ซึ่งในโครงงานนี้จะเลือก Aglet ในการพัฒนาเนื่องมาจาก

(1) Aglet เป็นเครื่องมือที่ใช้ในการพัฒนาเอเจนท์เคลื่อนที่ที่มีผู้ใช้มากสุด

(2) Aglet ยังมีการพัฒนาอยู่ในปัจจุบัน

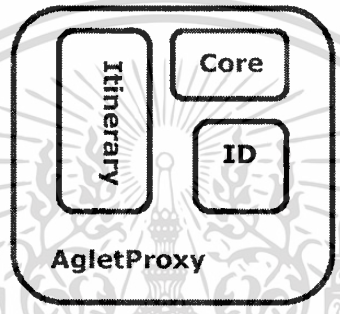
(3) Aglet ประกอบด้วยส่วนของเส้นทางที่ใช้ในการเดินทางของเอเจนท์เคลื่อนที่ ซึ่งในบางระบบไม่มี เช่น Telescript

(4) Aglet สามารถสร้าง facilities ที่มีความปลอดภัยให้แก่เอเจนท์ได้ เมื่อมีการเคลื่อนย้าย Aglet ไปยังเครื่องระยะไกล

Aglet Workbench ถูกพัฒนาขึ้นโดยบริษัท IBM มีเป้าหมายเพื่อใช้เป็นเครื่องมือในการสร้าง Stand alone Mobile agents ด้วยภาษาจาวา โดยในแพ็คเกจดังกล่าวจะประกอบด้วยเซิร์ฟเวอร์ของเอเจนท์ และโพรโทคอลที่ใช้ในการเคลื่อนย้ายเอเจนท์ (Agent Transfer Protocol)

กุญแจสำคัญเมื่อกล่าวถึงเทคโนโลยีเอเจนท์เคลื่อนที่ คือความสามารถในการเคลื่อนที่ ซึ่งหากพิจารณาโครงสร้างของ Aglet จะพบว่า ประกอบด้วยส่วนประกอบ 2 ส่วน ได้แก่ Core เป็น

เหมือนหัวใจของ Aglet ประกอบด้วย ตัวแปรและเมธอดภายใน Aglet หน้าที่ของ Core จะเป็นส่วนประสานที่ Aglet ใช้สื่อสารผ่านไปยังสิ่งแวดล้อมรอบๆตัว ในส่วนของ Core จะถูกห่อหุ้มด้วย Aglet proxy ซึ่ง Aglet proxy นี้จะทำหน้าที่เป็นเกราะป้องกันทุกสิ่งที่จะเข้าถึง Aglet โดยตรง และยังสามารถซ่อนที่อยู่ที่แท้จริงของ Aglet ได้ เป็นการป้องกันการทำลายจากผู้ไม่ประสงค์ดี นอกจากนี้ Aglet ยังประกอบด้วย Identifier ที่ไม่ซ้ำกัน และมีเส้นทางที่จะต้องเดินทางด้วย โครงสร้างของ Aglet เป็นดังรูปที่ 2.4



รูปที่ 2.4. โครงสร้างของ Aglet

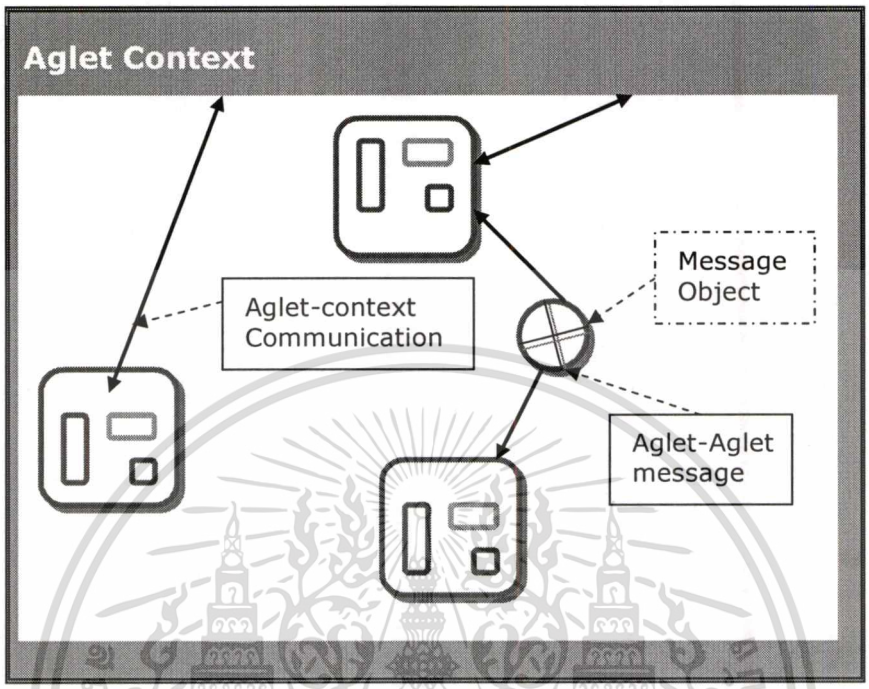
นอกจากโครงสร้างของ Aglet แล้ว คุณลักษณะอื่นของ Aglet ที่มีสำคัญส่วนได้แก่

2.3.1 เซิร์ฟเวอร์มีความยืดหยุ่นในการทำงาน

เซิร์ฟเวอร์ที่จะให้ Aglet เข้ามาทำงานได้จะแตกต่างจากเซิร์ฟเวอร์ในระบบ ไคลเอนท์/เซิร์ฟเวอร์คือ เอเจนท์เคลื่อนที่สามารถเปลี่ยนแปลงพฤติกรรมของเซิร์ฟเวอร์ได้เช่น เอเจนท์เคลื่อนที่อาจเข้าไปเพิ่มตารางในฐานข้อมูล หรือแก้ไขแฟ้มข้อมูลบางอย่างในเซิร์ฟเวอร์ได้

Aglet ต่างจากเอเจนท์เคลื่อนที่อื่นๆ เนื่องจากการเคลื่อนย้ายของ Aglet จะต้องทำงานในแม่ข่ายที่เข้ากันได้กับ Aglet เท่านั้น เซิร์ฟเวอร์ของ Aglet อาจเรียกว่า Context ก็ได้ หน้าที่ของ Context เป็นเสมือนคลังสินค้าหรือที่ทำงานของ Aglet ซึ่ง Aglet ทุกประเภทสามารถเข้ามาติดต่อสื่อสารกันได้ ดังรูปที่ 2.5

Context เป็นวัตถุที่อยู่กับที่ ไม่ได้เคลื่อนย้ายไปกับเอเจนท์ หน้าที่ของ Context คือให้บริการที่อยู่ และจัดการกับ Aglet ในสภาพแวดล้อมที่ปลอดภัยจาก Aglet ที่ประสงค์ร้าย การทำงานกับ Context Aglet จะได้มาซึ่งข้อมูลเกี่ยวกับสภาพแวดล้อม และ Aglet ก็สามารถรับ-ส่งข้อความกับสิ่งแวดล้อม และกับ Aglet อื่นๆได้



รูปที่ 2.5 สภาพแวดล้อมของ Aglet

2.3.2 การโต้ตอบเฉพาะที่

ประโยชน์ของเอเจนท์เคลื่อนที่ คือการเคลื่อนย้ายข้อมูลที่ต้องการ จากเซิร์ฟเวอร์เฉพาะที่ ซึ่งเอเจนท์สามารถตอบสนองความต้องการได้ทันท่วงที โดยไม่ต้องรอการส่งข้อมูลนานๆ Aglet ใช้การสื่อสารผ่านการส่งข้อความระหว่าง ตัว Aglet เองผ่าน Context กระบวนการนี้จะทำการแลกเปลี่ยนข้อความระหว่าง Aglet 2 ตัว และสามารถส่งข้อความถึงกันได้แบบประสานเวลาและไม่ประสานเวลากัน

2.3.3 ภาระเครือข่ายที่ลดลง

แนวคิดของ Aglet ช่วยลดภาระการทำงานของเครือข่ายได้ โดยการเคลื่อนย้าย Aglet ไปยังแม่ข่ายที่เหมาะสม เพื่อประมวลผล การเคลื่อนย้ายของ Aglet กระทำโดยอาศัย

(1) Agent Transfer Protocol (ATP)

(2) Java Agent Transfer and Communication Interface (J-ATCI)

ATP เป็นงานประยุกต์ในระดับโพรโทคอล สำหรับระบบการกระจายเอเจนท์ และอำนวยความสะดวกแก่ Aglet ในการเคลื่อนย้ายผ่านเครือข่าย บนพื้นฐานการใช้วิธีการตั้งชื่อ Naming

convention ของอินเทอร์เน็ต ATP ใช้ URL ในการระบุที่อยู่ของแม่ข่าย เป็นการรักษาความเป็นอิสระของแพลตฟอร์ม สำหรับการส่งผ่านเอเจนท์เคลื่อนที่ข้ามเครือข่าย

2.3.4 ภาวะอิสระ

ด้วยความสามารถในการตัดสินใจด้วยตนเองของเอเจนท์ บนพื้นฐานของสิ่งแวดล้อม เอเจนท์ได้รับอนุญาตให้นำเอางานที่จะต้องทำติดตัวไปด้วย ถึงแม้ว่าแม่ข่ายที่สร้างเอเจนท์ขึ้นมาจะไม่ได้ทำงานอยู่ในขณะนั้น

การโปรแกรม Aglet ด้วยการกำหนดเป้าหมายที่จะต้องทำให้สำเร็จ ทำได้โดยการใช้ Itinerary class ร่วมกับ Aglet (Itinerary object เปรียบเหมือนแผนการเดินทางของ Aglet) แล้วห่อหุ้มคำสั่งสำหรับ Aglet ที่ต้องการจะทำเมื่อไปยังแม่ข่าย ที่ต้องการ

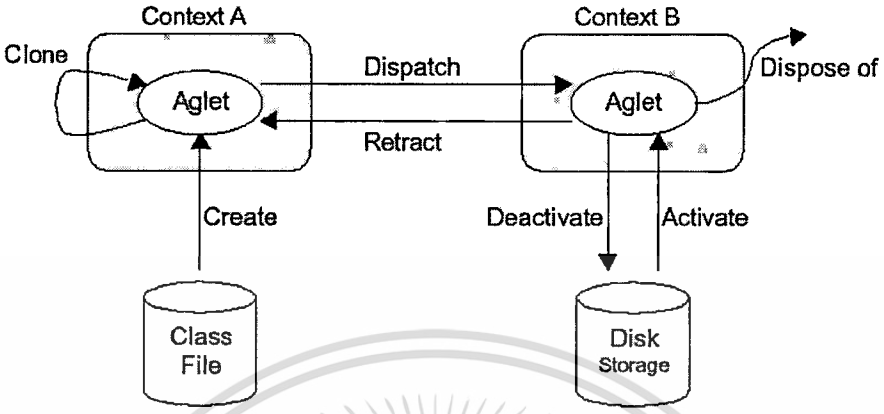
2.3.5 Aglet Life-Cycle Model

ตัวแบบ Aglet เป็นตัวแบบที่เรียกว่า Life and death of mobile agents ซึ่งการจะสร้าง Aglet มีด้วยกัน 2 วิธี คือ สร้าง Aglet ขึ้นใหม่ (Creation) หรือทำการคัดลอกจาก Aglet ที่มีอยู่ (Cloning) และเพื่อทำการควบคุมประชากร Aglet สามารถทำลาย Aglet ได้ (Disposal) Aglet เป็นการเคลื่อนที่แบบ 2 ทาง คือ Active และ Passive คุณลักษณะของการเคลื่อนที่แบบ Active คือ Aglet สามารถส่งตนเองจากแม่ข่ายหนึ่งๆ ให้ออกไปทำงานแบบระยะไกลได้ (Dispatching) ส่วนการเคลื่อนที่แบบ Passive คือคุณลักษณะที่แม่ข่ายระยะไกล ทำการดึง Aglet ออกจากแม่ข่ายปัจจุบันที่กำลังทำงานอยู่ (Retracting) และเมื่อ Aglet ทำงานอยู่โดยใช้ทรัพยากรบนแม่ข่าย เพื่อเป็นการลดการใช้ทรัพยากรของ Aglet Aglet สามารถหยุดพักได้ชั่วขณะหนึ่ง หรือทำการปล่อยทรัพยากรคืน (Deactivation) แล้วจึงกลับไปทำงานใหม่ใน Running mode (Activation) Aglet สามารถแลกเปลี่ยนข้อมูลกันเพื่อทำงานให้สำเร็จ

ที่กล่าวมาข้างต้น เป็นส่วนหนึ่งของการทำงานของ Aglet ที่จะทำเมื่อมีการสร้าง Aglet และจัดการกับ Aglet ที่อยู่กระจัดกระจายในสิ่งแวดล้อม ซึ่งสรุปเป็นการทำงานพื้นฐานของ Aglet ได้ดังรูป 2.6

Creation: คือการสร้าง Aglet ขึ้นมาใน Context การสร้าง Aglet นี้จะมีการกำหนดตัวระบุของ Aglet ก่อนจะถูกใส่เข้าไปใน Context แล้วจึงทำการเริ่มต้น

Cloning: การคัดลอก Aglet เป็นการคัดลอก Aglet จากต้นฉบับ ขึ้นมาใน Context ความแตกต่างของ Aglet ที่คัดลอกใหม่ กับต้นฉบับจะมีตัวระบุและการทำงานของ Aglet ใหม่จะเริ่มใหม่เพราะการคัดลอกนี้ไม่ได้คัดลอก Execution threads มาด้วย



รูปที่ 2.6 แบบจำลองวงจรชีวิต Aglet

Dispatching: การส่ง Aglet จาก Context หนึ่งไปยัง Context อื่น นั่นคือการเคลื่อนย้าย Aglet จาก Context ปัจจุบัน และทำการใส่ Aglet เข้าไปใน Context ปลายทาง ซึ่ง Context ปลายทางนี้จะทำการเริ่มการประมวลผลใหม่ เรียกการทำงานแบบนี้ว่าการ Push

Retraction: การ Retract ก็คือดึงหรือลบ Aglet ออกจาก Context ปัจจุบัน และใส่ Aglet เข้าไปใน Context ที่มีการร้องขอการ Retract

Activation and deactivation: การ Deactivation ของ Aglet คือความสามารถที่จะหยุดการประมวลผลชั่วคราว และเก็บสถานะของตนเองไว้ในหน่วยความจำ เช่น Disk Activation คือการส่งคืนตนเองกลับเข้าไปใน Context

Disposal: Aglet จะหยุดการประมวลผล และเอาตัวเองออกจาก Context ที่ทำงานอยู่

Messaging: การส่งข้อความระหว่าง Aglet ที่เกี่ยวข้องกับการส่ง รับ และจัดการกับข้อความ ทั้งแบบประสานเวลาและไม่ประสานเวลา

2.3.6 แบบจำลองเหตุการณ์ของ Aglet

การโปรแกรม Aglet อยู่บนพื้นฐานของแบบจำลองเหตุการณ์ แบบจำลองนี้ผู้พัฒนาโปรแกรมสามารถเพิ่มตัวสังเกตการณ์ (Listeners) เข้าไปใน Aglet ได้ ตัวสังเกตการณ์จะจับเหตุการณ์ที่เกิดขึ้นในวงจรชีวิตของ Aglet และอนุญาตให้ผู้พัฒนาโปรแกรมให้ Aglet กระทำบางสิ่งเมื่อ Aglet นั้นถูก Dispatch

ตัวสังเกตการณ์มี 3 ประเภทดังนี้

1. **Clone Listener** คอยสังเกตการณ์การโคลน เกิดเมื่อ Aglet ทำการโคลนเสร็จแล้ว

2. **Mobility Listener** คอยสังเกตการณ์การเคลื่อนที่ เกิดเมื่อ Aglet ถูก Dispatch ไปยัง Context อื่น เมื่อ Aglet ถูก Retract จาก Context และเมื่อ Aglet เดินทางไปถึง Context ใหม่แล้ว
3. **Persistence Listener** คอยสังเกตการณ์การมีอยู่ เกิดเมื่อ Aglet ถูก Deactivate และหลังจากที่ Activate แล้ว

CloneLimiter เป็นตัวอย่างการใช้ Listener ซึ่งสามารถนำ Listener นี้ไปใช้กับ Aglet ใดๆ ใหม่ได้ เรียกว่าเป็น Reusable listener

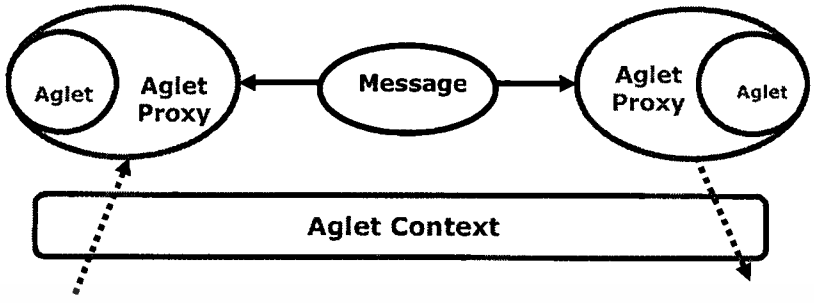
ตัวอย่างที่ 2.1

```
public class CloneLimiter implements CloneListener() {
final integer MAX = 5;
boolean original = true;
int number_of_clones = 0;
// Called when the aglet is about to be cloned.
public void onCloning(CloneEvent ev) {
if (original == false)
throw new SecurityException( "Clone cannot create a clone");
if (number_of_clones > MAX)
throw new SecurityException("Exceeds the limit");
}
// Called in the cloned aglet.
public void onClone(CloneEvent ev) {
original = false;
}
// Called in the original aglet after the cloning.
public void onCloned(CloneEvent ev) {
number_of_clone++;
}
}
```

จากตัวอย่างที่ 2.1 CloneLimiter จะนำ CloneListener มาใช้ การทำงานของโปรแกรมเพื่อจำกัดจำนวน Aglet ที่จะสร้างใหม่ด้วยการ Clone สำหรับ CloneListener มี 3 เมธอดคือ onCloning, onClone และ onCloned ซึ่งแต่ละเมธอดจะถูกเรียกตามลำดับ เมื่อแอพลิเคชันพยายามที่จะทำการโคลน การโคลนจะทำได้เพียง 5 ครั้งเท่านั้น และจะโคลนได้เฉพาะ Aglet ต้นฉบับเท่านั้น

2.3.7 Aglet API

Aglet API ระบุถึงการทำงานพื้นฐานของเอเจนท์เคลื่อนที่ รูปที่ 2.7 แสดงอินเตอร์เฟซและคลาสที่ถูกระบุอยู่ใน Aglet API รวมทั้งความสัมพันธ์ระหว่างอินเตอร์เฟซ



รูปที่ 2.7 Aglet API

Aglet API มีความยืดหยุ่นมาก เนื่องจาก Aglet API ถูกสร้างด้วยจาวา และนำมาใช้กับแนวคิดเอเจนท์เคลื่อนที่ Aglet API เป็น Java package ประกอบด้วยคลาสและอินเตอร์เฟซดังนี้ `Aglet`, `AgletProxy`, `AgletContext` และ `Message` คลาสเหล่านี้นักพัฒนาโปรแกรมสามารถใช้ในการสร้างและทำงานกับ Aglet ประกอบด้วยเมธอดที่ใช้สำหรับ Initialize Aglet จัดการกับข้อความ และทำการ Dispatch, Retract, Deactivate/Activate, Clone และ Dispose

2.3.8 Aglet Architecture

สถาปัตยกรรมของ Aglet มี 2 เลเยอร์ และ 2 APIs ซึ่งระบุถึงอินเตอร์เฟซสำหรับการเข้าถึงการทำงาน จากรูปที่ 2.8 Aglet runtime layer ทำงานเป็น Aglet API ระบุพฤติกรรมของ API component ได้แก่ AgletProxy และ AgletContext Aglet runtime layer มีการทำงานพื้นฐานสำหรับ Aglet ที่สร้างใหม่ จัดการกับ Aglet และ Dispatch ไปยังแม่ข่ายระยะไกล ซึ่งมีส่วนประกอบย่อยที่ถูกออกแบบมาเพื่อจัดการกับงานต่างๆดังนี้

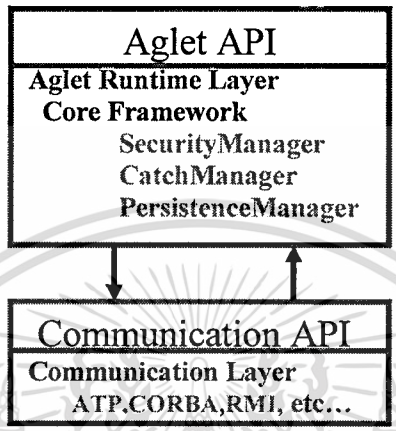
PersistenceManager รับผิดชอบในการเก็บเอเจนท์ลงไปในฮาร์ดดิสก์ ประกอบด้วย โค้ดและสถานะของ Aglet

CacheManager รับผิดชอบในการดูแลรหัสไบต์ที่ Aglet ใช้ เนื่องจากรหัสไบต์ของ Aglet ที่เดินทางเข้ามาจะต้องถูกเคลื่อนย้ายไปกับ Aglet

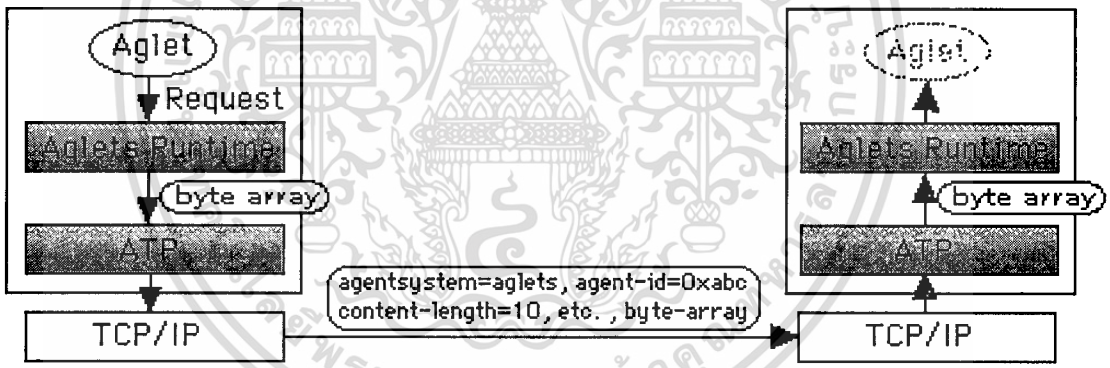
SecurityManager รับผิดชอบในการป้องกันแม่ข่ายและ Aglet จากเอนทิตีที่ประสงค์ร้าย SecurityManager จะจับทุกการทำงานที่มีผลต่อความปลอดภัยของระบบ และตรวจสอบว่าผู้ที่เรียกได้รับอนุญาตให้ทำงานนั้นหรือไม่

Communication layer จะเป็นชุดของเมธอดที่สนับสนุนการส่งข้อความและสถานะระหว่างกันของเอเจนท์ โดยไม่ขึ้นกับระบบเอเจนท์ทำให้ Runtime Layer เป็นอิสระกับกลไกการ

ส่ง OMG กำหนดมาตรฐานของ communication API สำหรับเอเจนท์ มีชื่อว่า MASIF (Mobile Agent System Interoperability Facility) ซึ่งทำให้ระบบเอเจนท์ต่างๆ สามารถทำงานร่วมกันได้ รับผิดชอบในการส่งผ่านเอเจนท์แบบเรียงลำดับให้เดินทางไปถึงปลายทาง



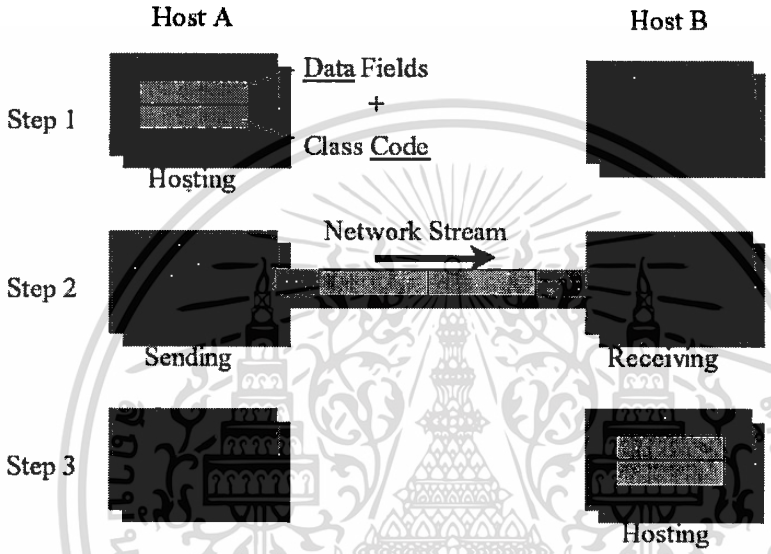
รูปที่ 2.8 Aglet architecture



รูปที่ 2.9 สถาปัตยกรรม Aglet

Aglet ใช้ Agent Transfer Protocol (ATP) ในการแทน Communication layer ATP เป็นตัวแบบที่สร้างมาจากโพรโทคอล HTTP (ใช้การเรียกชื่อแบบ URL) และเป็นโพรโทคอลในระดับ Application layer สำหรับการส่งผ่านเอเจนท์เคลื่อนที่ เพื่อสนับสนุนการสื่อสารของเอเจนท์แบบระยะไกล ATP จะรักษาความเป็นอิสระของ Platform สำหรับการส่งผ่านเอเจนท์เคลื่อนที่ข้ามเครือข่าย โดย ATP จะพยายามสร้างการเชื่อมต่อโดยตรงกับแม่ข่าย แต่ในเครือข่ายส่วนใหญ่ มักมี Fire wall ป้องกันผู้ใช้จากการเปิดการเชื่อมต่อผ่าน Socket ทำให้ Aglet ไม่สามารถ Dispatch หรือ Retract ผ่าน Fire wall ไปได้ จึงมีเทคนิคที่เรียกว่า HTTP Tunneling เกิดขึ้น ซึ่งจะทำให้ ATP สามารถส่งคำขอออกจาก Fire wall เป็นคำขอ HTTP *POST* และนำผลตอบกลับมาเป็น HTTP

ได้ จากรูปที่ 2.10 แสดงให้เห็นการส่งผ่าน Aglet ข้ามเครือข่ายไปยังเครื่องแม่ข่าย ซึ่งในที่นี้ Network Stream ก็คือ HTTP Tunneling นั่นเอง



รูปที่ 2.10 การถ่ายโอนของเอเจนท์

บทที่ 3
การออกแบบและพัฒนาระบบ

3.1 วัตถุประสงค์

เทคโนโลยีเอเจนท์เคลื่อนที่เป็นแนวคิดใหม่ ที่สามารถเพิ่มประสิทธิภาพให้กับงานในหลายๆด้าน เช่นเดียวกับงานด้านการเข้าถึงฐานข้อมูลในเครือข่าย การศึกษานี้มีวัตถุประสงค์ที่จะนำแนวคิดระบบเอเจนท์เคลื่อนที่มาพัฒนาโปรแกรมประยุกต์เอเจนท์เคลื่อนที่ กำหนดกรณีศึกษาเพื่อพัฒนาเอเจนท์ตั๋วโดยสารเครื่องบิน ทำหน้าที่คล้ายเอเจนท์ที่ผู้โดยสารไปติดต่อเมื่อต้องการซื้อตั๋วเครื่องบิน โดยที่เอเจนท์ตั๋วโดยสารจะได้รับข้อมูลจากผู้ใช้ เป็นข้อมูลสถานที่ที่ต้องการเดินทาง โดยระบุเมืองต้นทาง-เมืองปลายทาง สายการบินที่ต้องการใช้บริการ นอกจากนี้ มีข้อมูลอื่นๆ ได้แก่ เวลาออกเดินทาง เวลาที่ต้องการเดินทางไปถึง เป็นต้น ผลลัพธ์ที่ได้แสดงเป็นสายการบินราคาตั๋วโดยเรียงลำดับราคาตั๋วจากน้อยไปมาก

3.2 กำหนดปัญหาและแนวทางการแก้ไข

ในระบบการค้นคืนราคาตั๋วโดยสาร เมื่อผู้ใช้ต้องการค้นคืนราคาตั๋วโดยสาร จะต้องทำการค้นคืนราคาตั๋วโดยสารจากฐานข้อมูลของสายการบินต่างๆ แล้วนำราคาตั๋วโดยสารที่ได้มาเปรียบเทียบเพื่อหาราคาที่พอใจที่สุด ซึ่งเกิดความซ้ำซ้อนในการทำงาน เนื่องจากผู้ใช้ต้องป้อนข้อมูลต่างๆ (ต้นทาง-ปลายทาง วันที่และเวลาที่ต้องการจะเดินทาง ฯลฯ) ทุกครั้งที่ต้องการค้นคืนราคาตั๋วโดยสารของแต่ละสายการบิน เพื่อลดภาระการทำงานของผู้ใช้ในการค้นคืนราคาตั๋วโดยสาร และเพิ่มความสะดวกในการประมวลผลเปรียบเทียบข้อมูลราคาตั๋วโดยสารที่ต้องการ แนวคิดของระบบเอเจนท์ตั๋วโดยสารจึงถูกพัฒนาขึ้น โดยประยุกต์ใช้เทคโนโลยีเอเจนท์เคลื่อนที่เพื่อแก้ปัญหาการเข้าถึงฐานข้อมูลที่กระจัดกระจายของแต่ละสายการบิน

ในแง่ของการเข้าถึงฐานข้อมูล ระบบเอเจนท์ตั๋วโดยสารสามารถแก้ไขปัญหาความต่างกันของฐานข้อมูลได้ นั่นคือ เป็นไปได้ที่ Airline1 อาจใช้ฐานข้อมูล Oracle ในขณะที่ Airline2 อาจใช้ฐานข้อมูล SQL Server ซึ่งระบบเอเจนท์ตั๋วโดยสารสามารถเข้าถึงฐานข้อมูลที่ต่างกันเหล่านี้ได้

นอกจากนี้ สำหรับความคับคั่งในเครือข่าย ระบบเอเจนท์ตั๋วโดยสารจะช่วยลดความคับคั่งลงได้ เนื่องจากเอเจนท์ที่เคลื่อนที่ไปในเครือข่ายจะนำมาแต่ข้อมูลที่ผู้ใช้ต้องการเท่านลงได้

เนื่องจากเอเจนท์ที่เคลื่อนที่ไปในเครือข่ายจะนำมาแต่ข้อมูลที่ผู้ใช้ต้องการเท่านั้น ทำให้ช่องสัญญาณในเครือข่ายมีเพิ่มขึ้น

3.3 การออกแบบระบบ

ระบบเอเจนท์ตั๋วโดยสารมีหน้าที่ในการค้นคืนราคาตั๋วโดยสาร และนำเสนอข้อมูลราคาตั๋วโดยสารจากฐานข้อมูลของสายการบินต่างๆที่ผู้ใช้ระบุ ในการออกแบบระบบเอเจนท์ตั๋วโดยสารนี้จะขอกล่าวถึงภาพรวมการทำงานของระบบดังนี้

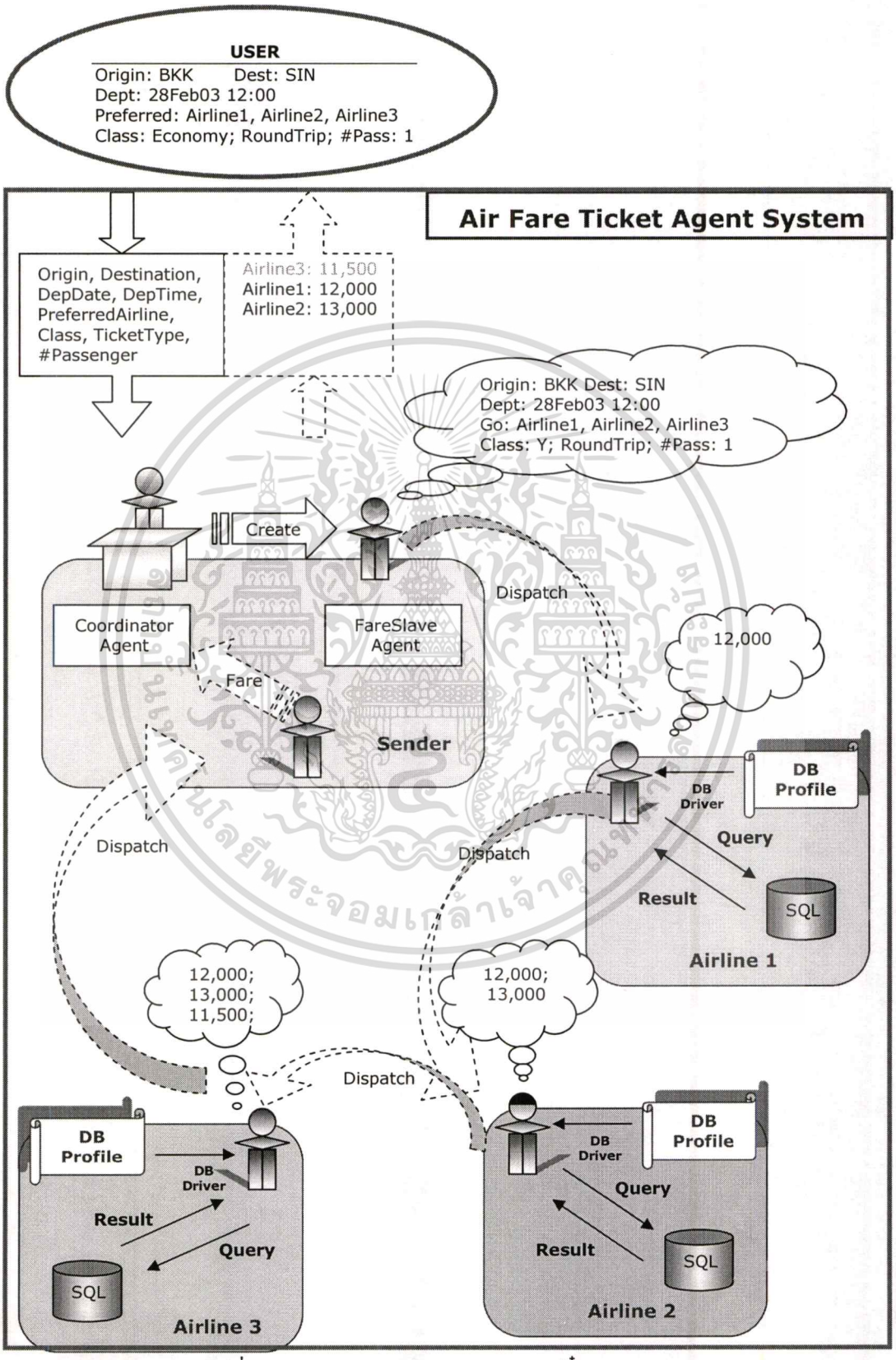
3.3.1 ภาพรวมการทำงานระบบ

การทำงานของระบบเอเจนท์ตั๋วโดยสารเริ่มจาก เมื่อผู้ใช้ต้องการค้นคืนราคาตั๋วโดยสารจากฐานข้อมูลของสายการบินต่างๆที่อยู่กระจัดกระจายในเครือข่าย ผู้ใช้จะเป็นผู้กำหนดเป้าหมายการทำงานของระบบเอเจนท์ตั๋วโดยสาร ภายในระบบเอเจนท์ตั๋วโดยสารประกอบด้วยเอเจนท์ 2 ตัว ทำงานร่วมกันอยู่ ได้แก่ เอเจนท์ Coordinator และเอเจนท์ Fare Slave ภาพรวมการทำงานของระบบแสดงให้เห็นดังรูปที่ 3.1

ในการทำงานร่วมกันของเอเจนท์ Coordinator และเอเจนท์ Fare Slave สามารถจำแนกหน้าที่การทำงานของเอเจนท์แต่ละตัวดังนี้

เอเจนท์ Coordinator เป็นเอเจนท์ไม่สามารถเคลื่อนที่ (Stationary agent) มีหน้าที่โต้ตอบกับผู้ใช้และเอเจนท์ Fare Slave เมื่อเอเจนท์ Coordinator ได้รับข้อมูล ได้แก่ เมืองต้นทาง เมืองปลายทาง วันที่และเวลาที่ต้องการเดินทาง สายการบินที่ต้องการ ฯลฯ ซึ่งเป็นข้อมูลที่ผู้ใช้กรอกผ่านหน้าจอ เอเจนท์ Coordinator จะนำข้อมูลที่ได้มากำหนดเป็นเป้าหมายและเส้นทาง แล้วสร้างเอเจนท์ Fare Slave ขึ้นมา นำเป้าหมายและเส้นทางการเดินทางให้แก่เอเจนท์ Fare Slave แล้วส่งเอเจนท์ Fare Slave ออกไปทำงาน จากนั้นเอเจนท์ Coordinator จะรอผลลัพธ์ที่เอเจนท์ Fare Slave จะนำกลับมาให้ (ระหว่างที่รอนี้เอเจนท์ Coordinator อาจสร้างเอเจนท์ Fare Slave ขึ้นมาทำงานอีกก็ได้ หรือเอเจนท์ Coordinator อาจตัดการเชื่อมต่อกับเครือข่ายชั่วคราวก็ได้ ทั้งนี้เพื่อลดความคับคั่งของเครือข่าย) เมื่อเอเจนท์ Fare Slave ทำงานเรียบร้อยจะนำผลลัพธ์ให้แก่เอเจนท์ Coordinator เพื่อนำผลลัพธ์นั้นมาประมวลผล โดยเรียงลำดับราคาตั๋วโดยสารจากน้อยไปมากและแสดงผลให้กับผู้ใช้

เอเจนท์ Fare Slave เมื่อได้รับมอบหมายงานและเส้นทางการเดินทางจากเอเจนท์ Coordinator การทำงานของเอเจนท์ Fare Slave เป็นดังรูปที่ 3.1 เมื่อเอเจนท์ Fare Slave ออกเดินทางไปยังโหนด Airline1 โดยนำเป้าหมายในการเดินทางไปด้วย ในที่นี้ก็คือตัวแปรที่เป็น

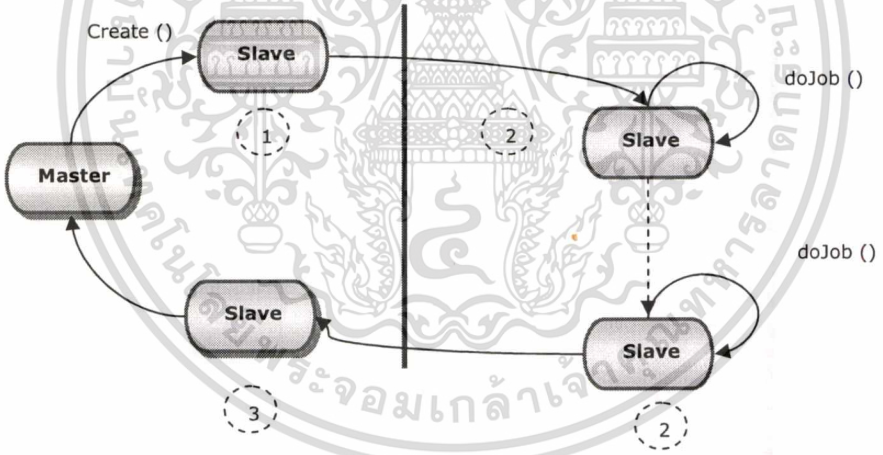


รูปที่ 3.1 ภาพรวมการทำงานของระบบเอเจนท์ตั๋วโดยสาร

เงื่อนไขในการสร้างข้อคำถามไปด้วย เมื่อเอเจนท์ Fare Slave เดินทางไปถึงจะเข้าไปอ่านข้อมูลการเข้าถึงฐานข้อมูลจากแฟ้มข้อมูล DB Profile ซึ่งประกอบด้วยข้อมูล ชื่อฐานข้อมูล ชื่อตารางที่เก็บราคาตั๋ว ประโยคเชื่อมต่อฐานข้อมูล (Connection string) เป็นต้น นำข้อมูลจาก DB Profile มาสร้างการเชื่อมต่อกับฐานข้อมูลเฉพาะที่ และใช้ในการสร้างข้อคำถาม ทำการเชื่อมต่อฐานข้อมูลและค้นคืนข้อมูลราคาตั๋วโดยสาร โดยใช้ข้อคำถามที่สร้างขึ้น เก็บผลลัพธ์ราคาตั๋วโดยสารของ Airline1 แล้วเดินทางต่อไปยัง Airline2, Airline3 ตามลำดับ แล้วจึงเดินทางกลับไปยังเครื่องผู้ส่งนำผลลัพธ์ทั้งหมดที่ได้มาส่งให้กับเอเจนท์ Coordinator เพื่อจัดเรียงข้อมูล และนำเสนอให้กับผู้ใช้

3.3.2 รูปแบบการทำงานของเอเจนท์

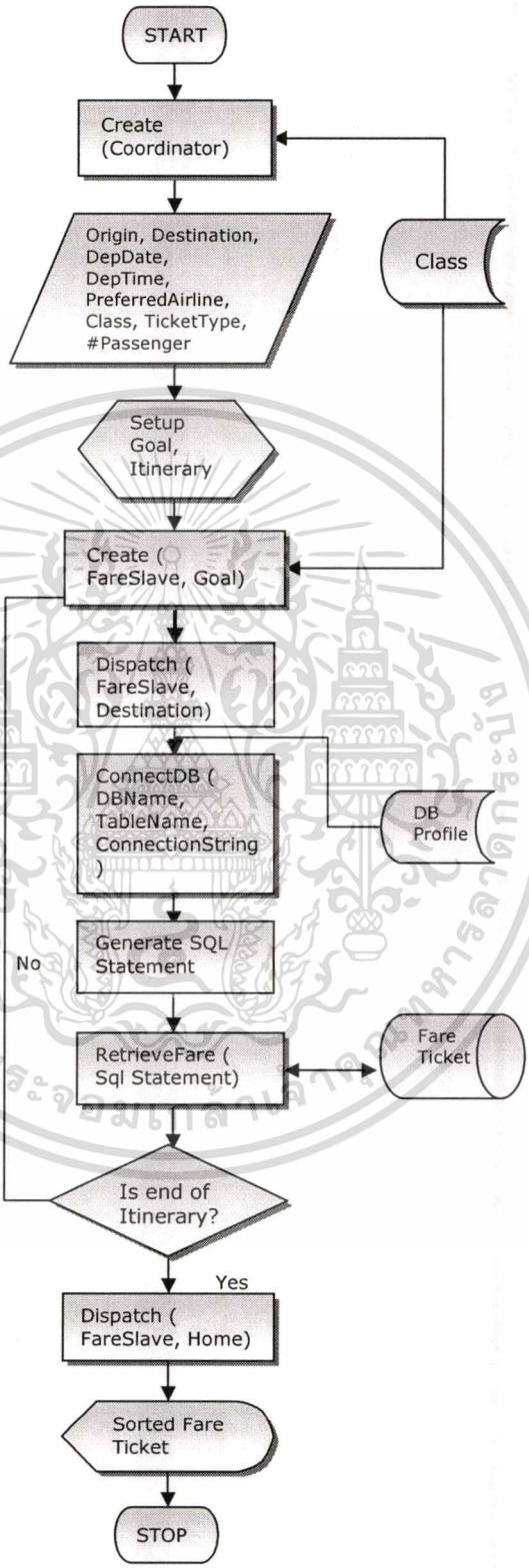
จากภาพรวมการทำงานทั้งหมดของระบบเอเจนท์ตั๋วโดยสาร เอเจนท์ที่ทำงานในระบบประกอบด้วย เอเจนท์ Coordinator และเอเจนท์ Fare Slave ซึ่งมีรูปแบบการทำงานร่วมกันแบบ Master – Slave ดังรูปที่ 3.2



รูปที่ 3.2 แบบจำลองการทำงานร่วมกันของเอเจนท์ Coordinator และ เอเจนท์ Fare Slave

จากแบบจำลองการทำงานของระบบเอเจนท์แบบ Master-Slave เอเจนท์ Coordinator จะไม่สามารถเคลื่อนที่ได้ มีหน้าที่มอบหมายงานให้แก่เอเจนท์ Fare Slave โดยทั้งเอเจนท์ Coordinator และเอเจนท์ Fare Slave ถูกพัฒนาบนสถาปัตยกรรมของ Aglet ทั้งในส่วนของโครงสร้างของเอเจนท์ การสื่อสารแบบส่งข้อความระหว่างกัน การเคลื่อนที่ผ่านเครือข่ายโดยใช้ ATP เป็นต้น (รายละเอียดหัวข้อ 2.3)

ระบบเอเจนท์ตั๋วโดยสารนี้ พัฒนาโดยใช้ภาษาจาวาซึ่งโครงสร้างการทำงานของโปรแกรมแสดงดังรูปที่ 3.3



รูปที่ 3.3 แผนผังโครงสร้างการทำงานของระบบเอเจนท์ตั๋วโดยสาร

ต่อไปจะเป็นการจำลองระบบฐานข้อมูล และสิ่งแวดล้อมของระบบเอเจนท์ตั๋วโดยสาร เพื่อแสดงให้เห็นการทำงานของระบบที่พัฒนาขึ้น

3.4 วิธีการจำลองสิ่งแวดล้อมของระบบ

จากภาพรวมการทำงานของระบบเอเจนท์ตั๋วโดยสารรูปที่ 3.1 สามารถแบ่งองค์ประกอบของระบบออกเป็นส่วนสำคัญต่างๆดังนี้

- **แพลตฟอร์ม** ในที่นี้คือเซิร์ฟเวอร์ หรือ Context ที่เอเจนท์จะเข้าไปทำงาน ซึ่งจะต้องประกอบด้วยสิ่งแวดล้อมที่เหมาะสม เพื่อให้เอเจนท์สามารถประมวลผลได้
- **ระบบเอเจนท์** สำหรับระบบเอเจนท์ตั๋วโดยสาร การทำงานของระบบจะประกอบด้วยเอเจนท์ 2 ตัวทำงานร่วมกัน ได้แก่ เอเจนท์ Coordinator และเอเจนท์ Fare Slave
- **ฐานข้อมูล** เป็นฐานข้อมูลแบบกระจาย ที่มีลักษณะแตกต่างกันและอยู่กระจัดกระจายในเครือข่าย

3.4.1 แพลตฟอร์ม (เซิร์ฟเวอร์)

ในปี 1997 OMG (The Object Management Group) ได้กำหนดมาตรฐานของเอเจนท์เคลื่อนที่ขึ้นเป็นครั้งแรก ระบุว่าเอเจนท์เคลื่อนที่ต้องมีแพลตฟอร์มที่สนับสนุน การประมวลผล การจัดการ การเคลื่อนย้ายตัวเอง ความปลอดภัย การติดสื่อสาร การทำรายการและการระบุเอเจนท์ ซึ่ง Aglet Workbench เป็นแพลตฟอร์มหนึ่งที่ถูกสร้างขึ้นตามมาตรฐานดังกล่าว

สำหรับการพัฒนาระบบเอเจนท์ตั๋วโดยสารซึ่งใช้แพลตฟอร์ม Aglet Workbench จะทำการฝัง Facility ลงไปที่เซิร์ฟเวอร์ที่จะให้ Aglet ทำงาน Facility เหล่านี้คือ API ที่เริ่มการทำงานของ Aglet เซิร์ฟเวอร์ที่เอเจนท์ตั๋วโดยสารจะทำงานได้จะต้องติดตั้ง Java Development Kit 1.4 และ ASDK (Aglet Software Development Kit) V.2.0.2 โดยติดตั้งทุกๆ เซิร์ฟเวอร์

<u>วิธีทำ</u>

1. ติดตั้ง ASDK V.2.0.2

2. เมื่อทำการติดตั้ง ASDK เรียบร้อยแล้วให้ทำการติดตั้งตัวแปรสิ่งแวดล้อม โดยสร้าง Startup script ที่จะใช้ในการติดตั้งตัวแปรที่เก็บ Path ของ JDK และ ASDK ซึ่งทำได้โดยการเพิ่มบรรทัดทั้งสองลงใน Autoexec.bat

```
SET AGLET_HOME  =  C:\Aglets-2.0.2\
SET JDK_HOME  =  C:\j2sdk1.4.0_02\
```

3. เรียกใช้ Startup Script file จากข้อ 2 โดยพิมพ์คำสั่ง autoexec.bat ที่ "C:\" ดังนี้

```
C:\autoexec.bat
```

4. เรียกใช้ Startup Script file ที่จะสร้างสภาพแวดล้อมที่เหมาะสมของเอเจนท์เซิร์ฟเวอร์ โดย เข้าไปที่ไดเร็คทอรี่ที่เราทำการติดตั้ง ASDK แล้วพิมพ์คำสั่งดังนี้

```
C:\Aglets-2.0.2\bin\agletsd.bat -port 4434
```

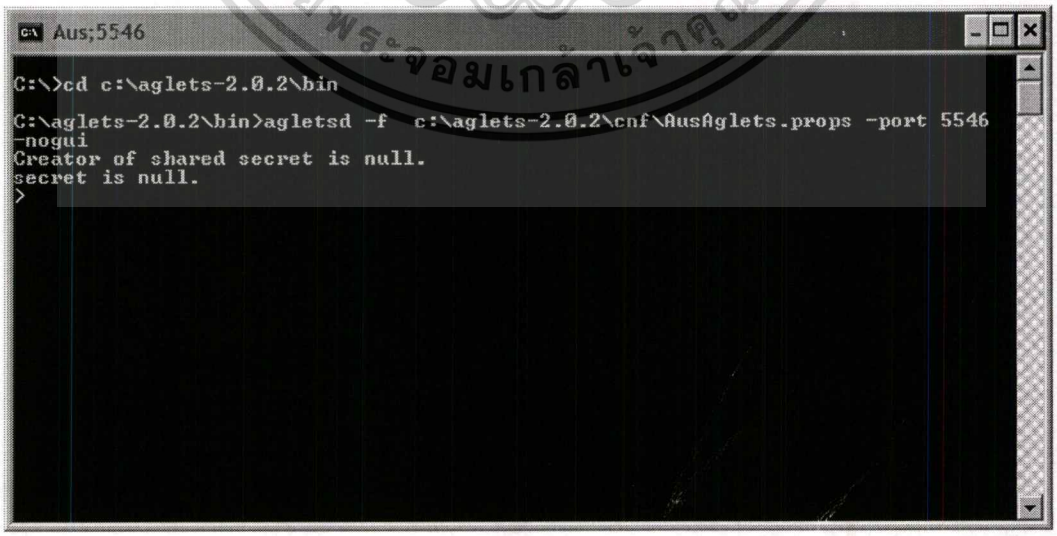
Agletsd.bat เป็น Startup Script ที่มีมากับ ASDK เมื่อเรียกใช้งาน จะเป็นการเริ่มโปรแกรมประยุกต์ที่อยู่ในระดับของ Aglet API โดยทำหน้าที่สร้างสภาพแวดล้อมที่เหมาะสม (Context) สำหรับให้เอเจนท์เข้ามาทำงาน ในที่นี้ ASDK ได้ให้ Startup script ดังกล่าวมากับแพ็กเกจแล้ว โดยผู้ใช้สามารถเรียกใช้งานได้ตามคำสั่งข้างต้น (`-port [number]` ใช้สำหรับการสร้าง Context ขึ้นมาโดยใช้ พอร์ตที่ระบุ การระบุพอร์ตจะเหมือนกับเราทำงานอยู่เครื่องหลายๆเครื่อง สามารถใช้ พอร์ตต่างกัน เมื่อต้องการสร้าง Context ขึ้นมาคนละพอร์ต เป็นการจำลองเสมือนกับทำงานอยู่บนเซิร์ฟเวอร์คนละเครื่อง)

สำหรับการศึกษานี้ จะใช้การระบุพอร์ตเพื่อจำลองการทำงานของเครือข่ายซึ่งกำหนดให้

PORT NUMBER : 4434 แทนเซิร์ฟเวอร์ของเอเจนท์ เป็นที่อยู่ของเอเจนท์ Coordinator และสามารถให้เอเจนท์เคลื่อนที่อื่นเดินทางมาประมวลผลได้

PORT NUMBER : 5546 (AUS), 5547 (USA) และ5548 (JAPAN) แทนเซิร์ฟเวอร์ของเอเจนท์ สมมติว่าเป็นเอเจนท์เซิร์ฟเวอร์ที่กระจายอยู่ในเครือข่าย

เมื่อการเรียกใช้ Startup Script ดังกล่าวเรียบร้อย จะปรากฏหน้าจอดังรูปที่ 3.4

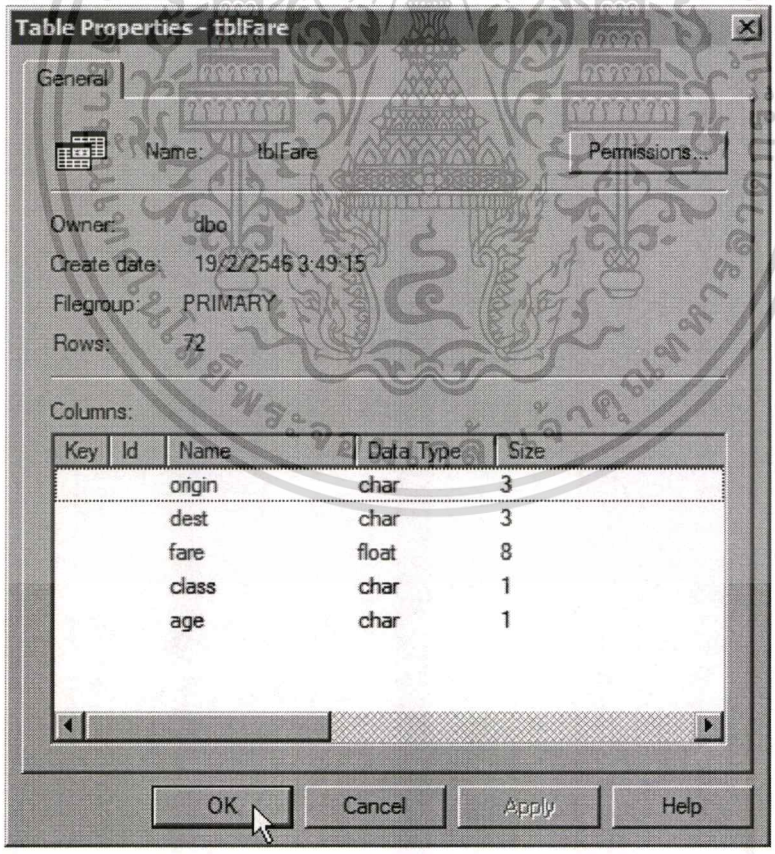


รูปที่ 3.4 หน้าจอเฝ้าคุมของเอเจนท์เซิร์ฟเวอร์

จากรูปที่ 3.4 แสดงหน้าจอเฝ้าคุมของเอเจนท์เซิร์ฟเวอร์ ใช้เป็นหน้าจอในการสร้าง การติดตาม หรือจัดการกับเอเจนท์เคลื่อนที่ และสามารถแสดงให้เห็นว่ามีเอเจนท์เคลื่อนที่ใดเข้ามาทำงานที่เซิร์ฟเวอร์ด้วย

3.4.2 จำลองระบบฐานข้อมูล

ระบบเอเจนท์ตั๋วโดยสารถูกพัฒนาขึ้นเพื่อทำงานกับฐานข้อมูลที่มีโครงสร้างแบบกระจาย ในการจำลองการทำงานของระบบเอเจนท์ตั๋วโดยสารจะทำการสร้างฐานข้อมูลเฉพาะที่ขึ้นในคอมพิวเตอร์ 3 เครื่องที่มีการเชื่อมต่อกันเป็นเครือข่าย แล้วสร้างฐานข้อมูลขึ้น 3 ส่วนบนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 (อาจใช้ระบบจัดการฐานข้อมูลที่แตกต่างกันเช่น Oracle) กำหนดโครงสร้างตารางจัดเก็บข้อมูลราคาตั๋วโดยสารดังรูปที่3.5 และตัวอย่างข้อมูลในตารางดังรูปที่ 3.6



รูปที่ 3.5 โครงสร้างตารางจัดเก็บข้อมูลราคาตั๋วโดยสาร

Data in Table 'tblFare'

origin	dest	fare	class	age
BKK	HKG	8420	A	1
BKK	HKG	8400	B	1
BKK	HKG	8300	C	1
BKK	HKG	8640	A	2
BKK	HKG	8630	B	2
BKK	HKG	8610	C	2
BKK	HKG	8520	A	3
BKK	HKG	8500	B	3
BKK	HKG	8480	C	3
BKK	HKG	8250	A	4
BKK	HKG	8240	B	4
BKK	HKG	8220	C	4
BKK	SIN	6150	A	1
BKK	SIN	6130	B	1
BKK	SIN	6110	C	1
BKK	SIN	6250	A	2
BKK	SIN	6230	B	2
BKK	SIN	6210	C	2
BKK	SIN	6350	A	3
BKK	SIN	6330	B	3
BKK	SIN	6310	C	3
BKK	SIN	6050	A	4
BKK	SIN	6030	B	4
BKK	SIN	6010	C	4
BKK	ZRH	30500	A	1
BKK	ZRH	30300	B	1
BKK	ZRH	30100	C	1
BKK	ZRH	30500	A	2
BKK	ZRH	30300	B	2
BKK	ZRH	30100	C	2
BKK	ZRH	30500	A	3
BKK	ZRH	30300	B	3
BKK	ZRH	30100	C	3
BKK	ZRH	30500	A	4
BKK	ZRH	30300	B	4
BKK	ZRH	30100	C	4
HKG	BKK	8420	A	1
HKG	BKK	8400	B	1
HKG	BKK	8300	C	1
HKG	BKK	8640	A	2

รูปที่ 3.6 ตัวอย่างข้อมูลในตารางราคาตั๋วโดยสาร

เมื่อสร้างฐานข้อมูลที่มีความแตกต่างกันแล้ว จะต้องสร้างแฟ้มข้อมูลเพื่อจัดเก็บข้อมูลของฐานข้อมูลเฉพาะที่นั้น ในแฟ้มข้อมูลดังกล่าวจะจัดเก็บข้อมูลทั่วไปของฐานข้อมูลเฉพาะที่ได้แก่ ชื่อฐานข้อมูล ไดร์เวอร์ที่ใช้เชื่อมต่อฐานข้อมูล ชื่อตารางที่เก็บข้อมูลราคาตั๋วโดยสาร และประโยคเชื่อมต่อฐานข้อมูลต่างๆดังนี้

แฟ้มข้อมูล ThaiAglet.props ให้ทำการจัดเก็บที่ C:\aglets_2.0.2\cnf ของคอมพิวเตอร์ที่จะเป็นตัวสร้างเอเจนท์ Coordinator

```
#(option) To run the Tahiti Server without security.
```

```
#the "aglets.secure" setting is set it to "false".
aglets.secure=true

# (optional) Which protocol to use(atp or rmi)
maf.protocol=atp

# (optional) Port number used by agents server.
maf.port=4434

# (optional) Resolve the domain name of the host by querying DNS
#server.
#atp.resolve=false
atp.resolve = true
# (optional) TCP/IP domain name of the host
atp.domain=Thai.com

# (optional) Set server's hostname to "localhost".  This is useful if
# the host does not have any network adapter.
atp.offline=true

# (optional) User servers ip address in server URL instead of
# logical name.  This is useful if you don't have a DNS e ntry.
atp.userip=true

# User ID for authorization.  This key must exist in your keystore.
# See keytool documentation for info on creating entry. (genkey)
aglets.owner.name=aglet_key

# Password for above user ID.  Must be same as entered as the key
#password
# used with keytool.
aglets.owner.password=aglets

# Keystore password.  Same as used with keytool.
aglets.keystore.password=aglets
```

แฟ้มข้อมูล AusAglet.props ให้ทำการจัดเก็บที่ C:\aglets_2.0.2\cnf ของคอมพิวเตอร์ AUS

```
#(option) To run the Tahiti Server without security.
#the "aglets.secure" setting is set it to "false".
aglets.secure=true

# (optional) Which protocol to use(atp or rmi)
maf.protocol=atp

# (optional) Port number used by agents server.
maf.port=5546

# (optional) Resolve the domain name of the host by querying DNS
#server.
atp.resolve = true
# (optional) TCP/IP domain name of the host
atp.domain=Thai.com

# (optional) Set server's hostname to "localhost".  This is useful if
# the host does not have any network adapter.
atp.offline=true
```

```
# (optional) User servers ip address in server URL instead of
# logical name.  This is useful if you don't have a DNS entry.
atp.userip=true

# User ID for authorization.  This key must exist in your keystore.
# See keytool documentation for info on creating entry. (genkey)
aglets.owner.name=aglet_key

# Password for above user ID.  Must be same as entered as the key
#password
# used with keytool.
aglets.owner.password=aglets

# Keystore password.  Same as used with keytool.
aglets.keystore.password=aglets

# list of all logical database names (separated by a colon) that can
#be accessed at the aglet server,
db.name= Aus
db.description="Sql Server"
db.driver=com.microsoft.jdbc.sqlserver.SQLServerDriver
db.url=jdbc:microsoft:sqlserver://latitude:1433;DataBaseName=Aus
```

แฟ้มข้อมูล UsaAglet.props ให้ทำการจัดเก็บที่ C:\aglets_2.0.2\cnf ของคอมพิวเตอร์ JAPAN

```
#(option) To run the  Tahiti Server without security.
#the "aglets.secure" setting is set it to "false".
aglets.secure=true

# (optional) Which protocol to use(atp or rmi)
maf.protocol=atp

# (optional) Port number used by agents server.
maf.port=5547

# (optional) Resolve the  domain name of the host by querying DNS
#server.
atp.resolve = true
# (optional) TCP/IP domain name of the host
atp.domain=Thai.com

# (optional) Set server's hostname to "localhost".  This is useful if
# the host does not have any network adapter.
atp.offline=true

# (optional) User servers ip address in server URL instead of
# logical name.  This is useful if you don't have a DNS entry.
atp.userip=true

# User ID for authorization.  This key must exist in your keystore.
# See keytool documentation for i nfo on creating entry. (genkey)
aglets.owner.name=aglet_key

# Password for above user ID.  Must be same as entered as the key
#password
# used with keytool.
aglets.owner.password=aglets
```

```
# Keystore password.  Same as used with keytool.
aglets.keystore.pas sword=aglets

# list of all logical database names (separated by a colon) that can
#be accessed at the aglet server,
db.name= Usa
db.description="Sql Server"
db.driver=com.microsoft.jdbc.sqlserver.SQLServerDriver
db.url=jdbc:microsoft:sqlserver://latitude :1433;DataBaseName=Usa
```

แฟ้มข้อมูล JapanAglet.props ให้ทำการจัดเก็บที่ C:\aglets_2.0.2\cnf ของคอมพิวเตอร์ JAPAN

```
#(option) To run the Tahiti Server without security.
#the "aglets.secure" setting is set it to "false".
aglets.secure=true

# (optional) Which  protocol to use(atp or rmi)
maf.protocol=atp

# (optional) Port number used by agents server.
maf.port=5548

# (optional) Resolve the domain name of the host by querying DNS
#server.
atp.resolve = true
# (optional) TCP/IP domain name of the host
atp.domain=Thai.com

# (optional) Set server's hostname to "localhost".  This is useful if
# the host does not have any network adapter.
atp.offline=true

# (optional) User servers ip address in server URL instead of
# logical name.  This is useful if you don't hav e a DNS entry.
atp.userip=true

# User ID for authorization.  This key must exist in your keystore.
# See keytool documentation for info on creating entry. (genkey)
aglets.owner.name=aglet_key

# Password for above user ID.  Must be same as entered as the  key
#password
# used with keytool.
aglets.owner.password=aglets

# Keystore password.  Same as used with keytool.
aglets.keystore.password=aglets

# list of all logical database names (separated by a colon) that can
#be accessed at the aglet server,
databases= Japan
db.FlightInfo,description="Sql Server"
db.FlightInfo.driver=com.microsoft.jdbc.sqlserver.SQLServerDriver
db.FlightInfo.url=jdbc:microsoft:sqlserver://latitude:1433;DataBaseNa
me=Japan
```

ในการเชื่อมต่อกับฐานข้อมูลผ่าน JDBC ในการศึกษานี้จะใช้การเชื่อมต่อกับฐานข้อมูลผ่าน JDBC 2 แบบแตกต่างกันบนฐานข้อมูลแต่ละที่ กำหนดให้

ATP://thai:4434 ใช้ JDBC ODBC bridge driver

ATP://aus:5546 ใช้ JDBC Native driver

ATP://usa:5547 ใช้ JDBC ODBC bridge driver

ATP://japan:5548 ใช้ JDBC Native driver

การทำงานของระบบเอเจนท์ตั๋วโดยสาร เอเจนท์ไม่ได้นำไดรเวอร์ในการเชื่อมต่อกับฐานข้อมูลติดตัวมาด้วย ไดร์เวอร์ที่ใช้จะเป็นไดร์เวอร์ของฐานข้อมูลที่เอเจนท์เดินทางไป ดังนั้นเอเจนท์ Fare Slave จึงสามารถทำงานกับฐานข้อมูลต่างกันได้ ในที่นี้สมมติว่าทุกๆโหนดใช้ Native JDBC ในการเชื่อมต่อ เมื่อดึงข้อมูลโดยใช้คำสั่ง SQL และได้ผลลัพธ์เป็นทะเบียนข้อมูล ข้อมูลที่ได้จะถูกใช้ในการปรับเปลี่ยนข้อคำถามของเอเจนท์ Fare Slave แล้วเดินทางไปยังโหนดต่อไปจนกระทั่งครบตามเส้นทางที่กำหนด แล้วก็นำผลลัพธ์ทั้งหมดกลับมาให้เอเจนท์ Coordinator เพื่อนำไปแสดงผลให้กับผู้ใช้

จะเห็นได้ว่าการทำงานของเอเจนท์ Coordinator และเอเจนท์ Fare Slave เป็นการทำงานในลักษณะของ อะซิงโครนัส หรือไม่ประสานเวลา คือเอเจนท์ Coordinator ไม่จำเป็นต้องติดต่อกับเอเจนท์ Fare Slave ตลอดเวลา ในระหว่างที่เอเจนท์ Fare Slave ทำงานอยู่ เอเจนท์ Coordinator อาจไม่ได้เชื่อมต่ออยู่ในเครือข่ายก็ได้

3.5 การใช้งานระบบเอเจนท์ตั๋วโดยสารที่พัฒนา

เมื่อติดตั้งส่วนประกอบที่จำเป็นของระบบได้แก่ แพลตฟอร์ม และระบบฐานข้อมูล การใช้งานระบบสามารถปฏิบัติดังนี้

(1) เรียกใช้งาน Startscript ของเอเจนท์เซิร์ฟเวอร์ ในการทดลองนี้ กำหนดให้เอเจนท์เซิร์ฟเวอร์ ที่จะเป็นผู้สร้างเอเจนท์ Coordinator ชื่อว่า THAI ใช้ URL atp://thai:4434 ส่วนเอเจนท์เซิร์ฟเวอร์ อื่นก็ทำเช่นเดียวกัน

(2) ที่ Command line ของเอเจนท์เซิร์ฟเวอร์: THAI เมื่อเอเจนท์ Coordinator ถูกสร้างแล้วจะปรากฏหน้าจอสำหรับผู้ใช้ดังรูปที่ 3.7

(3) ผู้ใช้กำหนดเงื่อนไขของตั๋วโดยสาร สมมติว่าผู้โดยสารต้องการเดินทางจากกรุงเทพ-สิงคโปร์

(4) ผู้โดยสารสามารถกำหนดได้เองว่า ต้องการตั๋วโดยสารของสายการบินใดบ้าง โดยเลือกที่ Preferred Airline

(5) กดปุ่ม GO แล้ว Coordinator จะสร้าง FareSlave ขึ้นมา โดยให้เงื่อนไขที่ผู้ใช้กำหนด และเส้นทางที่ FareSlave จะต้องเดินทางไป

(6) ผลลัพธ์ที่ได้จะแสดงออกมาดังรูปที่ 3.8

จากการใช้งานระบบ จะพบว่าการค้นหาข้อมูลราคาตั๋วโดยสารเครื่องบินนั้น ผู้ใช้ไม่จำเป็นต้องทราบถึงโครงสร้างการกระจายของฐานข้อมูล รวมทั้งความซับซ้อนในการค้นคืน ต่อไปจะแสดงให้เห็นการทำงานของเอเจนท์เคลื่อนที่ ว่าเดินทางอย่างไร และค้นคืนข้อมูลอย่างไร รวมทั้งข้อมูลที่ได้จากแต่ละ เซิร์ฟเวอร์ ที่เอเจนท์ใช้ในการปรับเปลี่ยน ข้อคำถาม รวมทั้ง การเปลี่ยนแปลง Driver ของเอเจนท์ตามโครงสร้างของฐานข้อมูลโดยอัตโนมัติ

3.6 การทำงานของเอเจนท์ตั๋วโดยสาร

การใช้งานระบบผู้ใช้ไม่จำเป็นต้องทราบถึงโครงสร้างการกระจายของฐานข้อมูล วิธีการเชื่อมต่อกับฐานข้อมูล และการได้มาซึ่งข้อมูลราคาตั๋วโดยสาร เนื่องจากมีเอเจนท์ทำหน้าที่เหล่านี้ให้

การทำงานของระบบเริ่มเมื่อมีการเรียกการทำงานของเอเจนท์เซิร์ฟเวอร์ Thai เอเจนท์ Coordinator จะถูกสร้างขึ้น และมีการแสดงหน้าจอเพื่อรับข้อมูลจากผู้ใช้ดังรูปที่ 3.8 ข้อมูลที่ผู้ใช้กรอกจะถูกใช้เป็นเป้าหมาย และเส้นทางการเดินทางของเอเจนท์ Fare Slave เมื่อเอเจนท์ Fare Slave ถูกส่งออกไปทำงานด้วยคำสั่ง Create จากรูปที่ 3.10 จะเห็นข้อความ

```
Create : FareSlave from atp://localhost:4434/
Dispatch : FareSlave to atp://latitude:5546
```

คำสั่ง Dispatch จะส่งเอเจนท์ FareSlave ออกไปที่เครื่อง atp://latitude:5546

ข้อความที่เอเจนท์ FareSlave แสดงเมื่อเดินทางมาถึงเอเจนท์ เซิร์ฟเวอร์ Aus:5546 แสดงดังรูปที่ 3.11

```
I'm here : atp://localhost:5546/
```

ในการอ่านแฟ้มข้อมูล AusAglets.props ได้ข้อมูลทั่วไปของฐานข้อมูลเฉพาะที่ ที่อยู่บนเซิร์ฟเวอร์ AUS:5546 เอเจนท์ Fare Slave แสดงข้อมูลที่อ่านได้ทางหน้าจอเฝ้าคุมดังนี้

```
DatabaseName :Aus
Driver : com.microsoft.jdbc.sqlserver.SQLServerDriver
Url: jdbc:Microsoft:sqlserver://latitude:1433:DataBaseName=Aus
```

ต่อไปนี้เป็นข้อความที่เอเจนท์แสดง เมื่อมีเอเจนท์เคลื่อนที่เดินทางเข้ามาประมวลผล

```
Receive : FareSlave from localhost:4434
```

ข้อความที่เอเจนท์เซิร์ฟเวอร์แสดงเมื่อเอเจนท์ Fare Slave ทำการเชื่อมต่อกับฐานข้อมูลสำเร็จ ในที่นี้ใช้การเชื่อมต่อแบบ JDBC Native

```
Database connected : com.microsoft.jdbc.sqlserver.SQLServerConnection
```

SQL Statement ที่เอเจนท์ FareSlave นำมาด้วย และใช้ในการค้นคืนข้อมูลในฐานข้อมูลเฉพาะที่ ได้เป็นผลลัพธ์ในบรรทัดต่อไป

```
Select fare from tblFare where origin = 'BKK' and dest = 'HKG':
I'll keep fare. = 5500
```

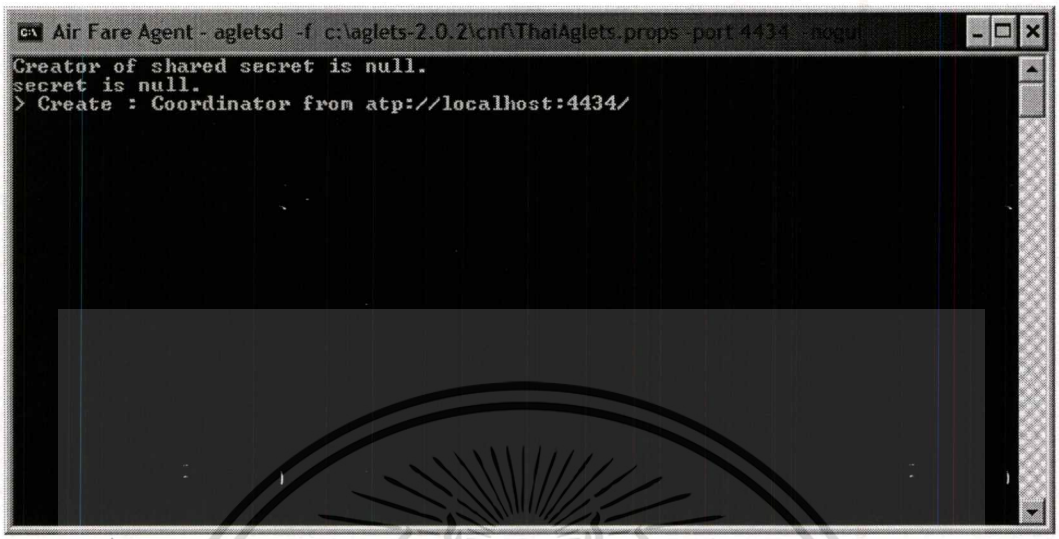
เมื่อทำงานเรียบร้อยแล้ว เอเจนท์ Fare Slave จะเดินทางต่อไปยังเอเจนท์เซิร์ฟเวอร์ ของสายการบิน: USA มี URL atp://latitude:5547 เอเจนท์จะแสดงข้อความจุดหมายต่อไปที่จะเดินทาง

```
Dispatch : FareSlave to atp://latitude:5547
```

การทำงานของเอเจนท์ FareSlave จะทำเช่นนี้เป็นวัฏจักร รูปที่ 3.12 และ 3.13 แสดงหน้าจอเฝ้าคุมของแต่ละเอเจนท์ เซิร์ฟเวอร์ ได้แก่ สายการบิน USA และสายการบิน JAPAN

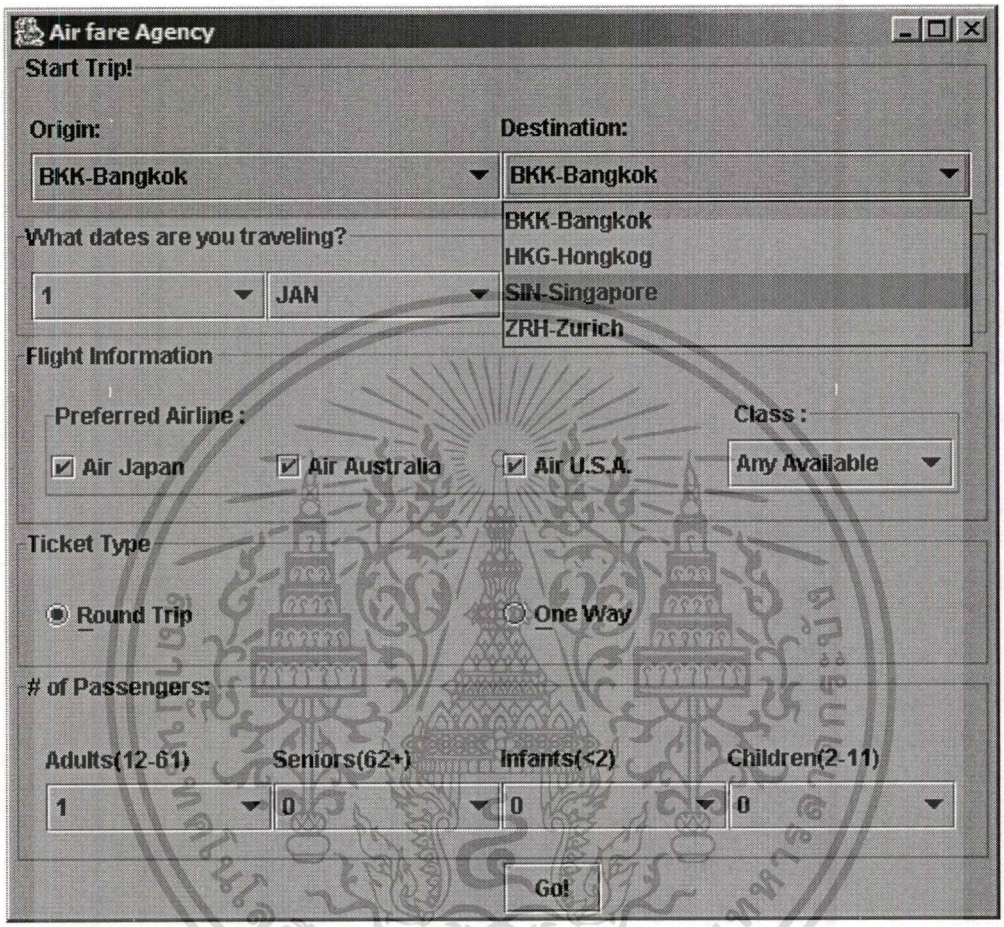
เมื่อเอเจนท์ Fare Slave เดินทางกลับมาที่ Host ซึ่งมี Coordinator ทำงานอยู่ เอเจนท์ FareSlave แสดงข้อความเมื่อเดินทางมาถึง แล้วส่งผลลัพธ์ที่ได้ให้เอเจนท์ Coordinator แสดงผลลัพธ์ดังรูปที่ 3.14

ระหว่างที่เอเจนท์ Fare Slave เดินทางไปในเครือข่าย เอเจนท์ Coordinator อาจทำการสร้างเอเจนท์ Fare Slave ขึ้นมาให้ทำงานอย่างเดียวกันก็ได้ โดยกำหนดเส้นทางการเดินทางไปยังเครื่องที่ต่างกัน เพื่อกระจายการทำงานของเครือข่าย แต่ทั้งนี้จะต้องเพิ่มฐานข้อมูลความรู้ (Knowledge base) สำหรับการตัดสินใจให้แก่เอเจนท์ Coordinator ด้วย

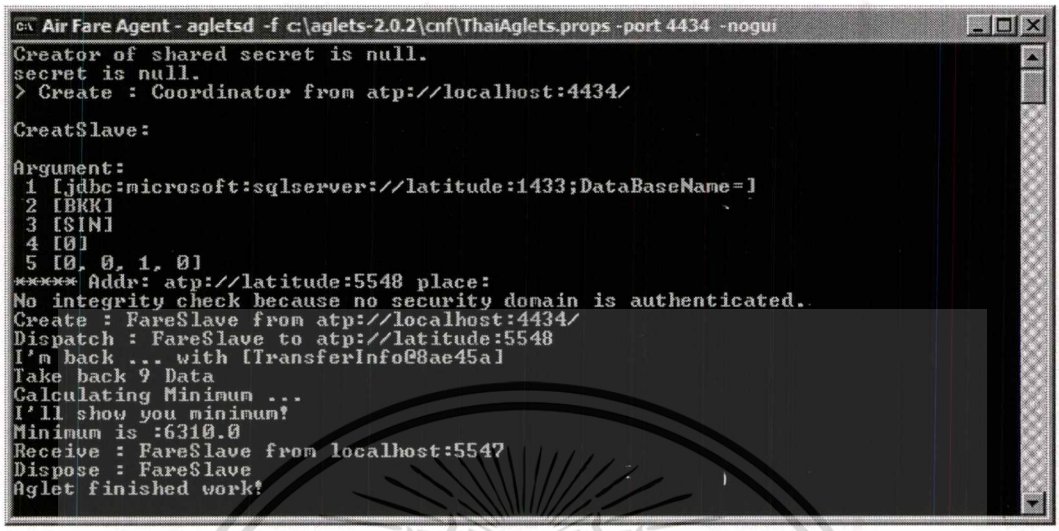


รูปที่ 3.7 หน้าจอแสดงการสร้างเอเจนท์ Master ชื่อ Coordinator

รูปที่ 3.8 หน้าจอรับข้อมูลจากผู้ใช้ของ Coordinator เอเจนท์

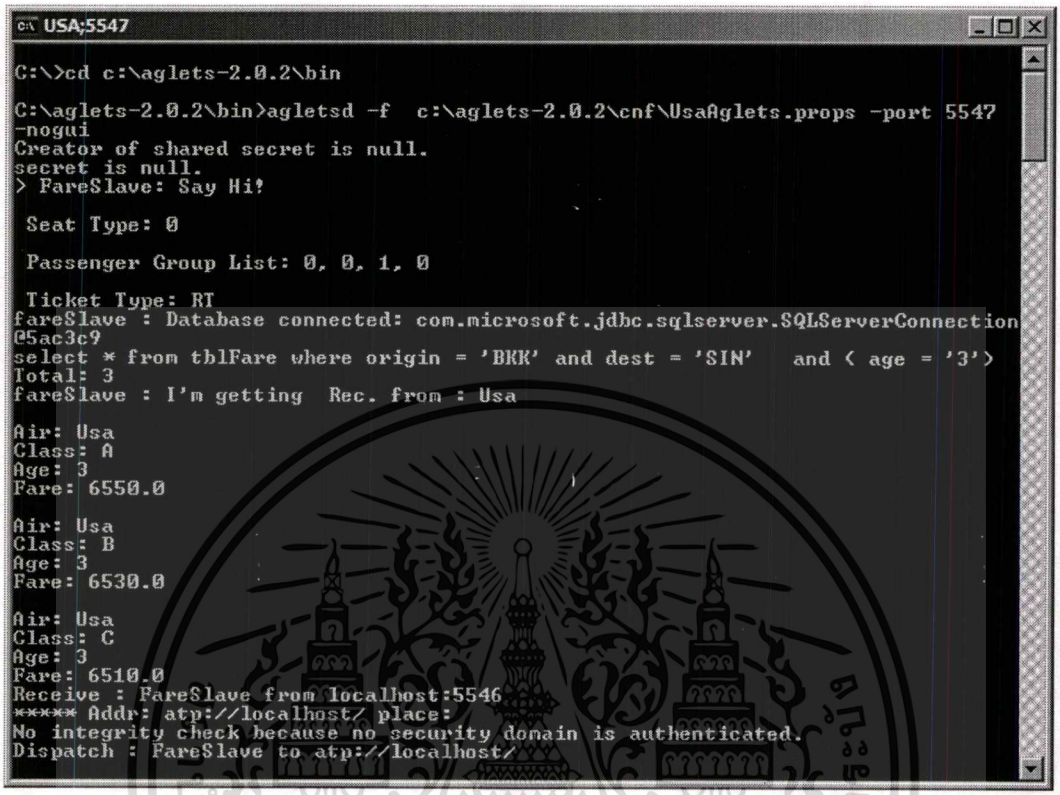


รูปที่ 3.9 ตัวอย่างหน้าจอสำหรับผู้ใช้ในการกำหนดเส้นทางการเดินทางของเอเจนท์เคลื่อนที่ เมื่อต้องการตั๋วเดินทางจากกรุงเทพไปสิงคโปร์

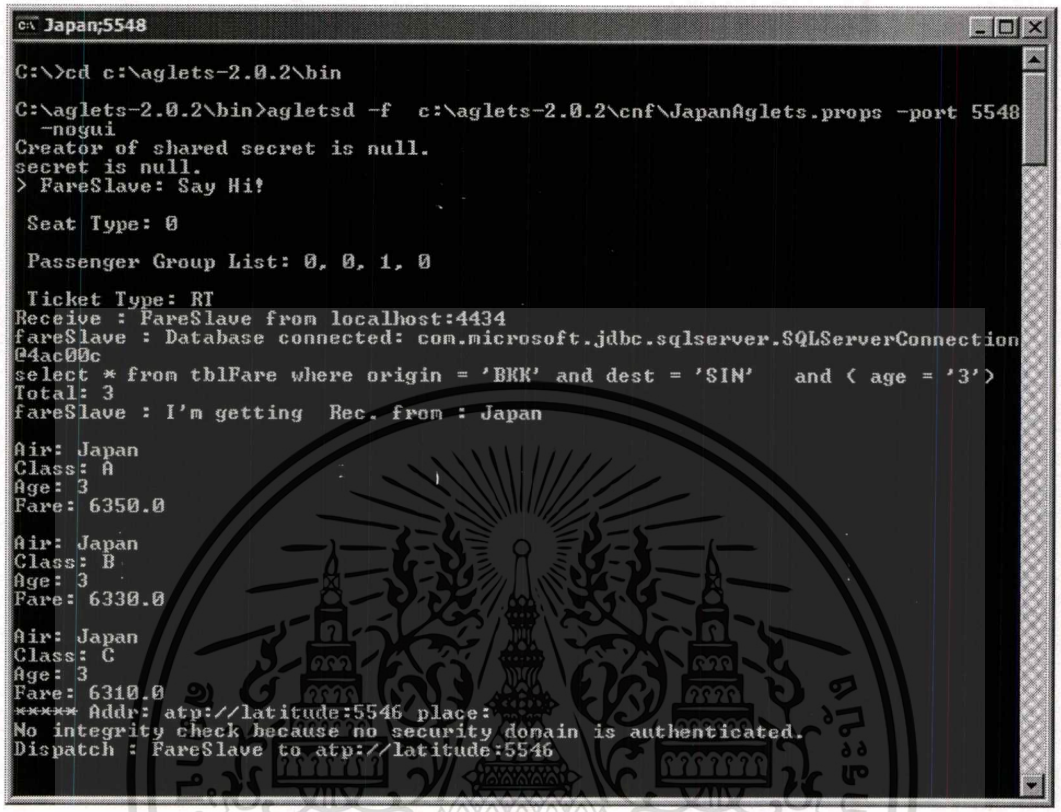


รูปที่ 3.10 หน้าจอการทำงานขอเอเจนท์ เซิร์ฟเวอร์ : Thai

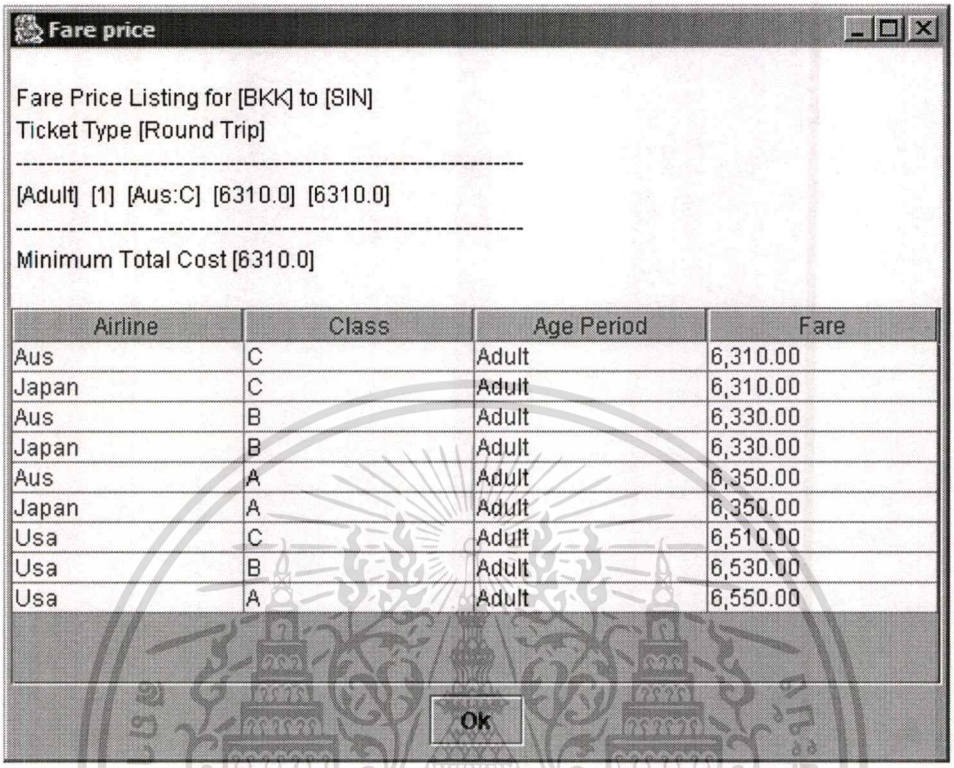
รูปที่ 3.11 หน้าจอการทำงานของเอเจนท์ Fare Slave เมื่อเดินทางไปถึงเซิร์ฟเวอร์ Aus:5546



รูปที่ 3.12 หน้าจอเฝ้าคุมของเอเจนท์เซิร์ฟเวอร์ USA



รูปที่ 3.13 หน้าจอเฝ้าคุมของเอเจนท์เซิร์ฟเวอร์ JAPAN



รูปที่ 3.14 หน้าจอแสดงราคาตั๋วโดยสารเรียงลำดับจากน้อยไปมาก

บทที่ 4
สรุปผลการศึกษา และข้อเสนอแนะ

4.1 สรุปผลการศึกษา

จากการพัฒนาระบบเอเจนท์ตั๋วโดยสาร ซึ่งใช้เทคโนโลยีเอเจนท์เคลื่อนที่ในการพัฒนาระบบ ประโยชน์โดยตรงที่ได้รับจากการทำงานของระบบในแง่การใช้งานของผู้ใช้ คือการลดภาระการทำงานที่ซ้ำซ้อนของผู้ใช้ในระบบ ช่วยให้ผู้ใช้ทำงานสะดวกรวดเร็วขึ้น ประโยชน์ที่ได้รับโดยอ้อม ได้แก่ การกระจายการทำงานของเครื่องคอมพิวเตอร์ในเครือข่าย ลดความคับคั่งในการใช้งานเครือข่าย เป็นระบบที่มีความยืดหยุ่นสูง เนื่องจากเป้าหมายในการทำงานของเอเจนท์สามารถปรับเปลี่ยนได้

ในการพัฒนาระบบเอเจนท์ตั๋วโดยสาร นอกจากความสามารถในการค้นคืนข้อมูลในโครงสร้างฐานข้อมูลแบบกระจายแล้ว อาจเพิ่มความสามารถของระบบในการจัดการฐานข้อมูลให้ทำการเพิ่ม หรือลด ระเบียนข้อมูลในฐานข้อมูลที่กระจายตัวได้ หรืออาจพัฒนาความสามารถของเอเจนท์ Coordinator ให้ทำงานแบบอัจฉริยะ โดยออกแบบฐานข้อมูลความรู้ให้แก่เอเจนท์ Coordinator ในการจัดกลุ่มฐานข้อมูลสายการบินตามทวีป และส่งเอเจนท์ Fare Slave ออกไปทำงานพร้อมๆกัน โดยกำหนดเป้าหมายและเส้นทางในการเดินทางต่างกันไป

ประเด็นปัญหาของระบบเอเจนท์ตั๋วโดยสาร เป็นประเด็นปัญหาเดียวกันกับที่ระบบเอเจนท์เคลื่อนที่ทั่วไปกำลังแก้ไข คือความปลอดภัยของ ซึ่งขณะนี้มีการค้นคว้าวิจัยในประเด็นปัญหาดังกล่าวอย่างแพร่หลาย เนื่องจากความปลอดภัยของระบบเป็นสิ่งที่ไม่อาจมองข้ามได้

บรรณานุกรม

Dale, J. 1995. "A Mobile Agent Architecture to Support Distributed Resource Information Management." PhD Thesis, Faculty of Engineering and Computer Science, University of Southampton.

Danny, B. Lange. and Mitsuru, O. 1998. **Programming and Deploying Java(TM) Mobile Agents with Aglets(TM).** Addison-Wesley.

Demartini, C., et al. 1999. "A DDR Based Approach for System Management" pp. 437-444. In **Proceedings of the Fieldbus Conference FeT'99 Conference.** September 23-24.

Edward, W. 1999. **Mobile Code and Security.** [Online]. Available: http://www.cigital.com/presentations/mobilecode/sld005.htm.

Friedemann, M. and Kotz, D. 2000. **Agent Systems, Mobile Agents, and Applications.** [Online]. Available: http://www.inf.ethz.ch/department/vs/publ/papers/Incs1882toc.pdf.

Mitsuru, O., et al. 1998. **Aglets Specification 1.1 Draft.** [Online]. Available: http://www.cs.uidaho.edu/~abdullah/fmca/aglet/Aglets.htm

Nikhil, K. 1997. **Agentos - A Java Based Mobile Agent System.** [Online]. Available: http://netresearch.ics.uci.edu/agentos/nkothari/.

Rothermel, K. and Schwehm, M. 1998. **Mobile Agents.** Encyclopedia of Computer Science and Technology. New York: M. Dekker.

Sahuguet, A. 1997. **About Agents and Databases.** [Online]. Available: http://www.cis.upenn.edu/ ~sahuguet/Agents/Agents_DB.pdf.