เอเจนท์เคลื่อนที่

Mobile Agents

มีนา รัตนสาครกุล[†] ดร. ภัทรชัย ลลิตโรจน์วงศ์[‡]

บทคัดย่อ

เอเจนท์ หรือซอฟท์แวร์เอเจนท์ เป็นซอฟท์แวร์ที่ทำงานหนึ่งๆแทนผู้ใช้ เอเจนท์เคลื่อนที่ เป็นซอฟท์แวร์เอเจนท์ที่สามารถเคลื่อนที่ผ่านเครือข่ายจากจุดหนึ่งไปยังอีกจุดหนึ่งโดยควบคุมการทำงานด้วยตัวเอง เอเจนท์เคลื่อนที่จะเคลื่อนย้ายโค้ดและสถานะของเอเจนท์ไปประมวลผลที่โหนดใดๆในเครือข่ายซึ่งเป็นการทำงานในลักษณะของการคำนวณแบบกระจาย บทความนี้จะนำเสนอแนวคิดและหลักการทำงานของเอเจนท์เคลื่อนที่ รวมถึงงานวิจัยที่เกี่ยวข้อง เพื่อเป็นแนวทางในการนำเทคโนโลยีดังกล่าวไปพัฒนาและออกแบบระบบเอเจนท์ต่อไป

Abstract

An agent is software performing user's task. A mobile agent is an enhanced agent capable of traveling autonomously from one node to another on a network. During migration, agent's code and state information is saved and transferred to the destination. At the destination node, the state of the agent is restored so as to allow the program to continue where it is left off. Such procedure can be described as "distributed computation." This article presents the concepts, paradigms and characteristics of mobile agents, and previous research on the application of mobile agent technology.

1. บทนำ

ในช่วงเวลาที่ผ่านมา การใช้เครือข่ายอินเทอร์เน็ตได้เติบโตขึ้นอย่างรวดเร็ว ทำให้เทคโนโลยีของสารสนเทศอิเล็กทรอนิกส์มีการพัฒนาขึ้นอย่างมาก เนื่องจากสารสนเทศจะกระจัดกระจายอยู่ทั่วไปในเครือข่าย ทำให้ต้องมีการบริหารจัดการสารสนเทศเหล่านี้ให้มีประสิทธิภาพมากขึ้น อันได้แก่ การจัดการ การค้นหา และการรวบรวมสารสนเทศ นอกจากนี้ยังพบอีกว่าเมื่อสารสนเทศอิเล็กทรอนิกส์อยู่ในเครือข่ายระดับโลกแล้ว ย่อมมีผู้ใช้จำนวนมากที่ต้องการใช้สารสนเทศร่วมกัน ดังนั้น ปัญหาในการจัดการข้อมูลที่ยากอยู่แล้วนั้นจึงเพิ่มขึ้นไปอีก

ด้วยเทคโนโลยีในปัจจุบันจึงเกิดเครื่องมือที่ช่วยในการค้นหาสารสนเทศ จัดการกลุ่มของสารสนเทศที่อยู่กระจัดกระจาย และรวบรวมกลุ่มของสารสนเทศเหล่านั้นเข้ากับทรัพยากรอื่น ตัวอย่างเครื่องมือที่สร้างขึ้นเพื่อการค้นหาสารสนเทศ ได้แก่ search engine ต่างๆ จะทำงานแทนผู้ใช้ในการค้นหาสารสนเทศที่อยู่กระจัดกระจาย ลักษณะงานของ search engine นั้น เป็นตัวอย่างหนึ่งของการทำงานที่เรียกว่า เอเจนท์ (agent) นั่นเอง

เอเจนท์ เป็นซอฟท์แวร์แบบหนึ่งที่ทำงานหนึ่งๆแทนผู้ใช้หรือเอนทิตีอื่น เพื่อทำการรวบรวม กรอง หรือประมวลผลข้อมูล โดยสามารถทำงานและตอบสนองต่อการเปลี่ยนแปลงของสภาพแวดล้อมได้เองโดยอัตโนมัติ และการทำงานนั้นจะต้องเสร็จสมบูรณ์ [1] บางครั้งเอเจนท์ก็ต้องอาศัยความร่วมมือจากเอเจนท์หลายๆตัว คุณลักษณะโดยทั่วไปของเอเจนท์คือ มีจุดมุ่งหมาย (intention) มีความฉลาด (intelligence) มีภาวะ

[†] นักศึกษาปริญญาโทคณะเทคโนโลยีสารสนเทศ สจล.

[‡] อาจารย์คณะเทคโนโลยีสารสนเทศ สจล.

อิสระ (autonomy) (คือสามารถทำงานให้สำเร็จได้ด้วยตัวเอง) และสามารถเชื่อถือได้ (believability) [2]

เนื่องจากเอเจนท์เป็นซอฟท์แวร์ จึงมีข้อจำกัดเช่นเดียวกับที่ซอฟท์แวร์ทั่วไปมี เช่น ความแตกต่างของแพลตฟอร์ม การใช้งานร่วมกันของระบบ การเคลื่อนที่ผ่านเครือข่าย เป็นต้น เพื่อแก้ปัญหาดังกล่าว จึงเกิดการนำแนวความคิดของ mobile code เข้ามารวมกับเอเจนท์

mobile code คือโค้ดที่เดินทางข้ามเครือข่าย และไปทำกระทำการที่เครื่องอื่น โดยสามารถทำงานอยู่บนแพลตฟอร์มที่หลากหลายได้ mobile code จะถูกเขียนขึ้นเพียงครั้งเดียวแล้วทำงานแบบอัตโนมัติ ภาษาที่ใช้ในการเขียน mobile code ได้แก่ Java, JavaScript, VBScript, ActiveX, Postscript และ Word macros [3]

เอเจนท์เคลื่อนที่ (mobile agent) รวมเอาแนวคิดของเอเจนท์ และ mobile code เข้าด้วยกัน และมีการทำงานแบบการคำนวณแบบกระจาย (distributed computation) จากคุณลักษณะที่โดดเด่น ทำให้เอเจนท์เคลื่อนที่เป็นที่สนใจ และมีการศึกษากันอย่างแพร่หลาย ตลอดจนการนำไปใช้งานในด้านต่างๆ เช่น การจัดการเครือข่าย การค้นหาสารสนเทศที่กระจัดกระจายในอินเทอร์เน็ต การทำงานเป็นผู้ช่วยส่วนบุคคล ตลอดจน เกม และอื่นๆอีกมาก ในบทความนี้ จะนำเสนอรูปแบบ แนวคิด โครงสร้าง และคุณลักษณะของเอเจนท์เคลื่อนที่ พร้อมกับนำเสนอผลงานการศึกษาที่เกี่ยวข้องกับเอเจนท์เคลื่อนที่ เพื่อเป็นแนวทางในการนำแนวคิดดังกล่าวไปใช้ต่อไป

2. แนวคิดพื้นฐานของเอเจนท์เคลื่อนที่

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

เอเจนท์เคลื่อนที่ คือโปรแกรมที่สามารถเดินทางข้ามเครือข่ายได้เองโดยอัตโนมัติ และเข้าไปทำงานบางอย่างในคอมพิวเตอร์อื่นที่อยู่ในเครือข่าย ซึ่งต่างจากการทำงานแบบระบบรับ-ให้บริการ (client/server) นั่นก็คือ ในการทำงานแบบ client/server เอนทิตีที่ใช้ในการประมวลผล (เช่น โค้ด) จะถูกติดตั้งอยู่ที่เครื่องๆหนึ่ง ซึ่งแต่ละเครื่องจะมีบทบาทของตนเองชัดเจน คือ server จะให้บริการ และมี client เรียกใช้บริการ การสื่อสารระหว่าง client และ server ใช้วิธีการส่งข้อความ (message passing) ซึ่งการทำงานนั้น ผู้เขียนโปรแกรมจะต้องกำหนด network address ของผู้รับได้ [2] ในขณะที่เอเจนท์เคลื่อนที่จะทำการเคลื่อนย้าย/คัดลอกโปรแกรมและสถานะของเอเจนท์ไปประมวลผลยัง server ทำให้การประมวลผลที่เกิดขึ้นอยู่ใกล้กับข้อมูลมาก สารสนเทศที่ได้ก็ไม่ต้องเคลื่อนย้ายกลับมาจาก server

สถานะของเอเจนท์ประกอบด้วย 2 ส่วน คือ [5]

(1) สถานะของข้อมูล (data state) ประกอบด้วยตัวแปรของเอเจนท์ในระดับ global และ instance

(2) สถานะของการกระทำการ (execution state) ประกอบด้วยตัวแปรของเอเจนท์ในระดับ local

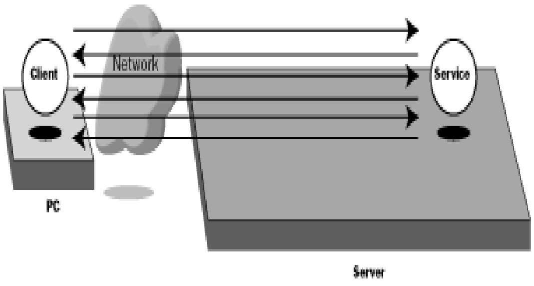
3. การเคลื่อนที่ของเอเจนท์เคลื่อนที่

การแบ่งแยกความแตกต่างของเอเจนท์เคลื่อนที่จะแบ่งตามระดับของการเคลื่อน ซึ่งแบ่งได้ 5 แบบด้วยกัน คือ [5]

(1) การส่งข้อความ (message passing) เป็นการสื่อสารระหว่างเอนทิตีในเครือข่าย โดยให้กระบวนการ (process) ทำการส่งและรับข้อความถึงกันโดยตรง แนวคิดนี้สนับสนุนการส่งข้อความแบบไม่ประสานเวลา ทำให้เกิดความยืดหยุ่นอย่างมาก เพราะสามารถรองรับการสื่อสารที่

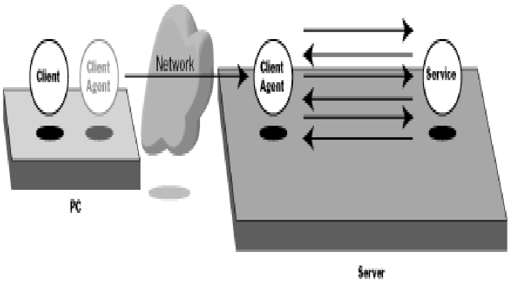
หลากหลายได้ แต่การพัฒนางานประยุกต์ที่ใช้การสื่อสารแบบนี้ ค่อนข้างที่จะซับซ้อน ทำให้วิธีนี้ไม่เป็นที่นิยม [1]

(2) remote procedure call (RPC) เป็นตัวแบบการสื่อสารกันระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ โดยที่คอมพิวเตอร์เครื่องหนึ่งสามารถเรียกใช้ procedure ของคอมพิวเตอร์อีกเครื่องหนึ่งได้ การเรียกใช้อาจมีการส่งข้อมูลไป เพื่อเป็น argument ของ procedure วิธีนี้การเรียกแต่ละครั้ง จะต้องมีการโต้ตอบกันถึง 2 ครั้งคือ การร้องขอของ client และการตอบรับจาก server ดังรูปที่ 1 แต่การทำงานแบบ RPC มีข้อจำกัดอย่างหนึ่งที่สำคัญมากคือ procedure ที่จะถูกเรียกจะต้องมีอยู่ที่เครื่องระยะไกล (remote node)



รูปที่ 1 Remote Procedure Call

(3) remote evaluation เป็นรูปแบบหนึ่งที่พัฒนาต่อจาก RPC คอมพิวเตอร์นอกจากจะสามารถเรียกใช้ procedure ของเครื่องอื่นแล้ว ยังสามารถเรียก/ส่ง โค้ดจากเครื่องอื่นมา/ไปทำงานได้ ดังในรูปที่ 2 การทำงานของคอมพิวเตอร์ทั้งสองเครื่อง จะทำงานตามคำสั่งที่อยู่ใน procedure แนวคิดนี้เป็นส่วนหนึ่งของแนวคิดที่เรียกว่า remote execution ซึ่งจะมีความสามารถในการเคลื่อนย้ายโค้ด (code mobility) แต่เอเจนท์เคลื่อนที่ต้องการความสามารถในการเคลื่อนย้ายกระบวนการ (process mobility) ซึ่งเป็นการโอนย้ายสถานะของข้อมูลและสถานะของการกระทำการ



รูปที่ 2 Remote Programming

(4) weak migration นอกจากการเคลื่อนย้ายโค้ดแล้ว เอเจนท์จะทำการเคลื่อนย้ายสถานะของข้อมูลไปด้วย โดยที่โปรแกรมเมอร์สามารถจำกัดขนาดของสถานะของข้อมูลนั้นๆได้

(5) strong migration เอเจนท์สามารถเคลื่อนย้ายโค้ดและสถานะ ทั้งในส่วนของข้อมูลและการกระทำการ ไปยังปลายทาง เมื่อเอเจนท์เดินทางไปถึงปลายทาง จะ restore สถานะ และทำงานต่อโดยอัตโนมัติ

4. คุณสมบัติของเอเจนท์เคลื่อนที่

ถึงแม้ว่า เราสามารถนำเอเจนท์เคลื่อนที่ไปประยุกต์ใช้ได้หลายทาง แต่เอเจนท์เคลื่อนที่จะต้องมีคุณสมบัติต่อไปนี้เหมือนกัน [2]

4.1 **การเคลื่อนย้าย** (migration) ความสามารถในการเคลื่อนที่ (mobility) เป็นคุณสมบัติที่เอเจนท์สามารถเคลื่อนที่ระหว่างโหนดในเครือข่ายได้ แต่การเคลื่อนย้าย (migration) คือการทำงานที่ควบคุมการเคลื่อนที่ให้สำเร็จลุล่วง การทำงานของเอเจนท์เคลื่อนที่เป็นกระบวนการปฏิบัติงานที่ทุกคำสั่งไม่จำเป็นต้องถูกประมวลผลที่โหนดใดเพียงโหนดเดียว ด้วยคุณสมบัติข้อนี้ ทำให้การเคลื่อนย้ายของเอเจนท์แตกต่างจากการเคลื่อนย้ายของกระบวนการ (process migration) โดยที่การเคลื่อนย้ายของกระบวนการจะทำโดยระบบ ซึ่งขึ้นอยู่กับที่อยู่ของทรัพยากร และการกระจายงานให้ได้ดุล (load balancing) ในขณะที่การเคลื่อนย้ายของเอเจนท์ ผู้ที่ตัดสินใจคือตัวเอเจนท์เอง แต่จะเคลื่อนที่ไปที่ใดนั้นก็ขึ้นอยู่กับสถานะ

ของเอเจนท์ และการเคลื่อนย้ายนั้น จะต้องมี infrastructure ที่สนับสนุนการร้องขอการประมวลผลนี้ด้วย

การเคลื่อนย้ายของเอเจนท์ระหว่างระหว่างโหนดสามารถแบ่งออกเป็น 2 แบบ คือ

(1) state-oriented เอเจนท์สามารถใช้คำสั่งในการเคลื่อนที่ (mobility imperative) เช่น go, jump หรือ move เพื่อเคลื่อนย้ายตัวเองได้ เมื่อคำสั่งเหล่านี้ถูกกระทำ สถานะของเอเจนท์จะถูกห่อหุ้มไว้ และถ่ายโอนผ่านเครือข่ายไปยังโหนดผู้รับ เมื่อโหนดผู้รับได้รับกระบวนการจากเอเจนท์ จะทำตามคำสั่งที่อยู่ถัดจากคำสั่งในการเคลื่อนที่นั้น การจับสถานะของเอเจนท์สามารถทำได้โดยตัวเอเจนท์เอง หรือไม่ก็โดย infrastructure

(2) stateless เอเจนท์สามารถเคลื่อนที่ไปยังโหนดต่างๆได้ แต่เมื่อย้ายไปแล้ว จะต้องทำงานใหม่ตั้งแต่ต้น ไม่ได้ทำหลังจากคำสั่งย้าย และก่อนที่เอเจนท์จะเคลื่อนย้าย ข้อมูลของสถานะจะถูกบันทึกในฐานความรู้ หรือไม่ก็รวบรวมไว้ด้วยกัน แล้วทำการห่อหุ้มและถ่ายโอนไปกับเอเจนท์ เมื่อเดินทางไปถึงโหนดผู้รับ เอเจนท์จะ restore สถานะเหล่านั้นจากที่เก็บ วิธีนี้จะเป็นการง่ายสำหรับเอเจนท์ในการเคลื่อนย้ายในระบบ แต่ยากที่จะเขียนตัวเอเจนท์ เพราะว่าต้องมีการ store/restore สถานะของเอเจนท์

4.2 การได้มาของข้อมูล เอเจนท์จะซักถามข้อมูลจากสิ่งแวดล้อมของตน (local environment) เพื่อให้ได้ข้อมูลที่จำเป็นในการทำงาน โดยที่เอเจนท์จะต้องกรอง และเลือกข้อมูลเหล่านี้ ก่อนที่จะส่งต่อไปยังโหนดเดิมของเอเจนท์

4.3 การกำหนดเส้นทาง หลังจากที่เอเจนท์ทำงานที่โหนดใดๆเรียบร้อยแล้ว จะต้องตัดสินใจว่า จะเดินทางไปยังโหนดใดต่อไป ที่มาของข้อมูลที่จะใช้การตัดสินใจนี้ มีได้หลายทาง [2] ได้แก่

(1) predetermination เอเจนท์จะถูกกำหนดปลายทางที่จะไปตั้งแต่เริ่มการทำงานแล้ว วิธีนี้ใช้เมื่อต้องการควบคุมการทำงานของเอเจนท์ เช่น เมื่อเราให้ความสำคัญกับลำดับของโหนดที่จะเข้าถึง

(2) dynamic determination เอเจนท์มีสิทธิที่จะไปยังโหนดใดๆก็ได้ในเครือข่าย อาจใช้การสุ่ม แต่เป็นการยากที่จะรู้ว่าโหนดใดในเครือข่ายที่สามารถเข้าถึงได้ อาจใช้ node knowledge history ของตนเอง หรือใช้ข้อมูลที่ได้จากเอเจนท์อื่น วิธีนี้มีประโยชน์อย่างมากต่อการทำ data mining เพราะต้องการข้อมูลจำนวนมากจากโหนดต่างๆ

(3) hybrid determination วิธีนี้เป็นการรวมสองวิธีข้างต้นเข้าด้วยกัน คือ จะมีการกำหนดรายการของโหนดที่จะเข้าถึงและไม่เข้าถึง แต่เอเจนท์ยังคงมีอิสระในการเลือก ถ้าโหนดนั้นตรงกับเป้าหมายที่ต้องการ นอกจากนี้ เอเจนท์ยังมีเกณฑ์การพิจารณาประเภทของโหนดที่จะเข้าถึงได้ หากโหนดใดไม่ตรงกับเกณฑ์ก็จะไม่ถูกเข้าถึง

5. ระบบเอเจนท์เคลื่อนที่

แบบจำลองของเอเจนท์จะอยู่บนแนวคิดของ “เอเจนท์และสถานที่” [5] จากรูปที่ 3 จะพบว่า ระบบเอเจนท์ (agent system) จะอยู่บนสถานที่หนึ่ง (place) ซึ่งเป็นที่ที่มีบริการต่างๆมากมายแตกต่างกัน

5.1 เอเจนท์ เอเจนท์เป็นเอนทิตีที่อาจเคลื่อนที่จากสถานที่หนึ่งไปยังสถานที่อื่น เพื่อพบกับเอเจนท์ตัวอื่น หรือเดินทางไปใช้บริการในสถานที่อื่น แบ่งเป็น 2 แบบคือ เอเจนท์เคลื่อนที่ และเอเจนท์บริการ (service agent) เอเจนท์บริการเป็นเอเจนท์ที่อยู่กับที่ ทำหน้าที่ให้บริการในสถานที่ต่างๆ บริการที่มีแบ่งเป็น 2 ระดับคือ (1) system service จะให้บริการแฟ้มข้อมูล และการเข้าถึงข้อมูลใน directory และ (2) application service เช่น บริการจองห้องพัก บริการสั่งดอกไม้ เป็นต้น ในทางเทคนิคแล้ว เอเจนท์บริการจะแปลงคำขอบริการจากเอเจนท์อื่นให้เป็น “ภาษาเอเจนท์” (agent language) เพื่อให้บริการเฉพาะบุคคลได้

5.2 สถานที่ สถานที่จะให้สภาพแวดล้อมที่ปลอดภัยแก่เอเจนท์ในการประมวลผล โดยทั่วไปแล้ว สถานที่หนึ่งก็คือโหนดหนึ่งของเครือข่ายนั่นเอง แต่ถ้าทุกบริการต่างก็อยู่บนโหนดเดียวกัน โหนดดังกล่าวก็จะจัดว่าเป็น multiple

place ตัวอย่างเช่น มีโหนดอยู่โหนดหนึ่งซึ่งถูกใช้แทนสถานที่หลายๆที่ ซึ่งสถานที่บนโหนดแต่ละแห่งถูกกำหนดให้เป็นของเอเจนท์แต่ละกลุ่ม การเข้าถึงบริการจะต้องใช้ policy ซึ่งไม่ใช่ว่าทุกความต้องการในสถานที่บนโหนดใดๆ จะต้องมีสถานที่เป็นของตนเอง เมื่อเป็นเช่นนี้ จึงเกิดเอเจนท์ที่ให้บริการสถานที่ โดยให้บริการทั้งแบบเฉพาะที่ (local) และระยะไกล (remote)ได้ด้วย

ความสามารถในการสื่อสารเป็นพื้นฐานของเอเจนท์เคลื่อนที่ ซึ่งมีเกณฑ์ที่ใช้ในการแบ่งหลายแบบด้วยกัน แบบแรกเป็นการแบ่งตามความสามารถในการสื่อสารข้ามเครือข่าย ประกอบด้วย [2]

(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 [2]

(1) ประสานเวลา ผู้ที่ทำการสื่อสารกันต้องจัดแบ่งเวลาที่จะใช้ในการสื่อสาร มีการประสานเวลาก่อนที่จะทำการส่งข้อมูล และมีการยืนยันการหลังจากการโอนย้ายข้อมูล วิธีนี้มักใช้เมื่อการโอนย้ายข้อมูลเป็นเรื่องสำคัญ ตัวอย่างเช่น เอเจนท์ที่ต้องการจะขอข้อมูลในฐานความรู้ของผู้อื่น

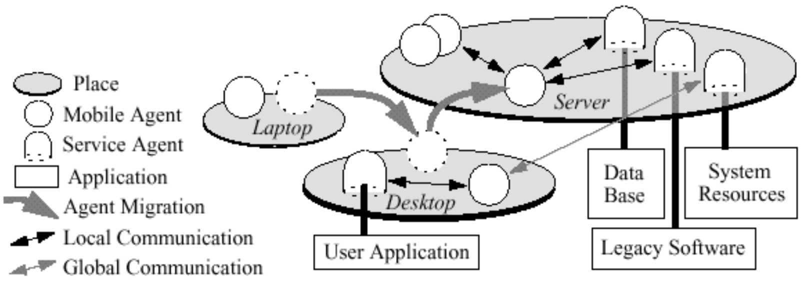
(2) ไม่ประสานเวลา ผู้ที่จะทำการสื่อสารกันสามารถที่จะติดต่อเมื่อใดก็ได้ตามต้องการ วิธีนี้มักจะใช้กับการโอนย้ายข้อมูลที่เป็นพวกข่าว ผู้รับต้องต้องตรวจสอบข้อมูลที่ได้รับนั้นเอง

5.3 การโต้ตอบ (interaction) ลักษณะของการโต้ตอบของเอเจนท์ในระบบ สามารถแบ่งได้ดังนี้ [5]

(1) เอเจนท์กับเอเจนท์บริการ เป็นลักษณะของ client/server นั่นคือ เอเจนท์บริการจะทำงานตามที่มีการร้องขอจากเอเจนท์อื่น มักจะใช้ RPC เป็นรูปแบบการสื่อสาร

(2) เอเจนท์เคลื่อนที่กับเอเจนท์เคลื่อนที่ การโต้ตอบประเภทนี้ต่างจากแบบแรก เพราะเอเจนท์ทั้งสองจะสื่อสารกันในลักษณะของ peer-to-peer มากกว่าที่จะเป็นแบบ client/server เอเจนท์แต่ละตัวจะมีวิธีควบคุมการโต้ตอบเพื่อบรรลุวัตถุประสงค์ของตัวเองตามต้องการ รูปแบบการสื่อสารที่ใช้ไม่ได้จำกัดอยู่ที่ RPC อย่างเดียว แต่จะขึ้นกับระดับความยืดหยุ่นที่ต้องการมากกว่า จึงอาจจะใช้การส่งข้อความก็ได้

(3) การสื่อสารของกลุ่มเอเจนท์ ในสองแบบแรกเราสมมุติว่าเอเจนท์ที่ทำการโต้ตอบกันนั้นรู้จักกัน นั่นคือ ผู้ส่งข้อความหรือ RPC จะรู้ว่าผู้รับข้อความของตนคือใคร แต่ถ้าผู้ส่งข้อความไม่รู้จักผู้รับ เช่น มีงานอยู่งานหนึ่งที่ให้เอเจนท์กลุ่มหนึ่งเป็นผู้ทำ แต่ละเอเจนท์ในกลุ่มก็รับงานย่อยๆ ไปทำงาน เอเจนท์แต่ละตัวสามารถสร้างกลุ่มของเอเจนท์ย่อยๆ ขึ้นมาอีก สมมุติว่ามีเอเจนท์บางตัวต้องการจบการทำงานของกลุ่มของตัวเอง เอเจนท์จะต้องส่งคำร้องขอเพื่อขอจบการทำงานออกไป ทั้งที่ไม่รู้จักสมาชิกแต่ละตัวในกลุ่มของตัวเอง เพราะเอเจนท์ที่ส่งรู้จักเพียงแต่กลุ่ม แต่ไม่รู้จักตัวเอเจนท์ใน



รูปที่ 3 ระบบเอเจนท์เคลื่อนที่

กลุ่ม การสื่อสารประเภทนี้จะต้องใช้ group communication protocol (เช่น ISIS toolkit) ซึ่งผู้ส่งจะส่ง event message ออกไปโดยที่ไม่รู้จักใคร ส่วนผู้รับก็จะแสดงตัวเองต่อ event ที่มันสนใจ

(4) ผู้ใช้กับเอเจนท์ เอเจนท์เคลื่อนที่ต้องนำผลลัพธ์ที่ได้กลับมาให้ผู้ใช้ โดยทั่วไปแล้วเอเจนท์เคลื่อนที่จะลงบันทึกข้อมูล (log) ไว้กับเอเจนท์ที่ทำงานอยู่บนโหนดก่อน แล้วจึงนำเสนอหรือประมวลผลข้อมูล ซึ่งการรวมข้อมูลและจัดรูปแบบให้ผู้ใช้ มักจะกระทำโดยเอเจนท์ตัวอื่น เอเจนท์ประเภทนี้มักรวมอยู่กับ desktop environment ของผู้ใช้ เช่น RTF ใน MS Word [2]

เมื่อเอเจนท์ต้องการสื่อสาร จะต้องสร้างความสัมพันธ์ในการสื่อสารที่เรียกว่า “session” เพื่อใช้ในการประสานเวลากันของเอเจนท์ที่ต้องการจะพบ (meet) และทำงานร่วมกัน เราสามารถแบ่ง session ได้เป็น 2 ประเภท คือ(1) ช่วงเวลาสื่อสารที่ใช้ภายในสถานที่เดียวกัน (intra-place session) คือ การสื่อสารที่เกิดขึ้นเฉพาะที่ (local) เท่านั้นและ (2) ช่วงเวลาสื่อสารที่ใช้ต่างสถานที่กัน (Inter-place session) คือ การสื่อสารครอบคลุม (global)

6. จุดเด่นของเอเจนท์เคลื่อนที่

เราสามารถนำเอาแนวคิดของเอเจนท์เคลื่อนที่ไปใช้ให้เกิดประโยชน์ เนื่องจากเอเจนท์เคลื่อนที่มีคุณสมบัติที่เป็นข้อดี ดังนี้ [7]

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

นอกจากนี้ การทำงานแบบประสานเวลายังมีประโยชน์ต่อการสร้างเส้นทางสื่อสารได้ เนื่องจากการสื่อสารที่เป็น session-based จะบังคับให้ต้องมีการเชื่อมต่อระหว่างผู้ส่งและผู้รับเกิดขึ้นอยู่ตลอดเวลา เอเจนท์จะเก็บประวัติของกระบวนการที่ทำงาน และสร้างการเชื่อมต่อขึ้น แม้ว่าจะไม่มีการทำงานใดๆก็ตาม

6.2 **การทำงานด้วยภาวะอิสระ** (autonomy) การทำงานด้วยภาวะอิสระ หมายถึง การทำงานให้สำเร็จได้ด้วยตัวเอง เอเจนท์เคลื่อนที่สามารถทำงานได้ด้วยตัวเอง โดยไม่ต้องการการเชื่อมต่ออยู่ตลอดเวลา เอเจนท์เคลื่อนที่จะนำเอาโค้ดของตัวเองไป และใช้ความรู้ที่มีตัดสินใจการทำงานเอง ซึ่งความรู้บางส่วนอาจจะได้จาก master ด้วยก็ได้ ทำให้ช่วยประหยัดเวลาในการเดินทางเป็นอย่างมาก

6.3 **สถาปัตยกรรมของเอเจนท์เคลื่อนที่** เนื่องจากสถาปัตยกรรมของเอเจนท์เคลื่อนที่เป็นแบบ decentralized, distributed และ light ทำให้การแก้ไขโค้ดของเอเจนท์ซึ่งมีขนาดเล็กเป็นไปได้ง่าย และการทำงานร่วมกันของโปรแกรมขนาดเล็กจะมีประโยชน์ต่อการจัดกำหนดการการทำงานของกระบวนการขนาดใหญ่ ให้สามารถใช้ทรัพยากรได้อย่างคุ้มค่า เพราะสถานีงาน (workstation) สามารถทำหน้าที่เป็น agent server ในขณะที่ไม่ได้ทำงานอะไร โดยไม่ได้ก่อให้เกิดค่าใช้จ่ายเพิ่มขึ้น

6.4 **ความสะดวกในการใช้เครื่องระยะไกล** เอเจนท์เคลื่อนที่ได้ประโยชน์จากการทำงานระยะไกล ดังนี้

(1) ซีพียู: การใช้ซีพียูของเครื่องระยะไกล ช่วยให้อุปกรณ์เคลื่อนที่ที่มีซีพียูจำกัด (thin client) สามารถทำงานได้มากขึ้น

(2) หน่วยความจำ: ในการทำงานที่ต้องใช้หน่วยความจำมาก การเข้าถึงหน่วยความจำของเครื่องระยะไกลจะช่วยแก้ปัญหาได้

(3) ระบบหลายตัวประมวล: ถ้าเครื่องระยะไกลนั้นมีหน่วยประมวลผลหลายตัว เอเจนท์ก็จะสามารถทำการประมวลผลแบบขนานได้

(4) แบนด์วิดท์: สามารถส่งเอเจนท์ไปทำงานในเครือข่ายที่มีช่องสัญญาณกว้างกว่า

(5) ทรัพยากรอื่นๆ: เอเจนท์อื่นสามารถใช้ทรัพยากรเฉพาะที่ได้โดยการควบคุมระยะไกล แต่ต้องระวังเรื่องความปลอดภัยของข้อมูลด้วย

แม้ว่าในทางทฤษฎี เอเจนท์จะมีข้อดีอย่างมากในเรื่องของการแบ่งปันการใช้ทรัพยากรให้คุ้มค่า แต่ก็ต้องแลกกับค่าใช้จ่ายในเรื่องของความปลอดภัยในการดูแลข้อมูลจากเอเจนท์ด้วยเช่นกัน

7. การประยุกต์ใช้งานเอเจนท์เคลื่อนที่

ก่อนที่จะออกแบบระบบ หรือนำรูปแบบของเอเจนท์เคลื่อนที่ไปใช้ ควรจะทำความเข้าใจในเรื่องของสถาปัตยกรรมและความต้องการของเอเจนท์เคลื่อนที่ ซึ่งมีรายละเอียดดังต่อไปนี้

7.1 สถาปัตยกรรม สถาปัตยกรรมโดยรวมของเอเจนท์เคลื่อนที่ขึ้นอยู่กับเครื่อง server ที่ติดต่อด้วยนั้น server ควรจะให้สภาพแวดล้อมในการทำงานที่เหมาะสมแก่เอเจนท์เพื่อให้เอเจนท์สามารถใช้ทรัพยากรที่นี่ได้

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

ขณะที่เดินทาง เอเจนท์จะไม่สามารถทำอะไรได้ แต่เมื่อเดินทางถึง server แล้ว เอเจนท์จะถูก activate ให้กระทำการ (execute) เหมือนกับเป็น local program ทั้งนี้ต้องมีข้อกำหนดในเรื่องของการรักษาความปลอดภัยด้วย ที่จุดนัดพบของเอเจนท์ (agent meeting point - AMP) server จะให้ทุกอย่างที่เอเจนท์ต้องการ โดยทั่วไปจะประกอบด้วยซีพียู หน่วยความจำ หน่วยเก็บ และข้อมูลเฉพาะที่ (local information)

7.2 สิ่งที่เอเจนท์เคลื่อนที่ต้องการ การนำเอเจนท์เคลื่อนที่ไปใช้ จำเป็นต้องมีสิ่งต่างๆต่อไปนี้ [7]

(1) ภาษาที่ใช้ได้หลายระบบ (portable language): โค้ดของเอเจนท์เคลื่อนที่ต้องสามารถทำงานได้ทุกที่โดยไม่ต้องแปล (compile) ใหม่ เนื่องจากจาวาเป็นภาษาที่สร้างขึ้นเพื่อทำงานบนแพลตฟอร์มที่หลากหลายได้ เพราะมีเครื่องเสมือน (virtual machine) และยิ่งกว่านั้น รหัสไบต์ (byte-code) ของจาวาก็ทำให้ภาษาโปรแกรมอื่นๆสามารถใช้จาวาในการพัฒนาได้ง่าย ดังนั้น จึงมีการนำจาวามาใช้ในการพัฒนาเอเจนท์เคลื่อนที่

(2) interpreted language: ตัวแปลคำสั่งจะทำหน้าที่เป็น virtual machine และในการแปลคำสั่งserver สามารถตรวจสอบโค้ดก่อนที่จะกระทำการได้ด้วย

(3) การตรวจสอบ (introspection): โปรแกรมควรมีความสามารถในการตรวจสอบอ็อบเจกต์ได้ โดยที่อ็อบเจกต์จะมี interface ให้โปรแกรมทำการตรวจสอบตัวมันด้วยเช่นกัน สิ่งนี้จะช่วยให้ server รู้ว่าอ็อบเจกต์นั้นกำลังจะใช้ทรัพยากรอะไร หรือทำให้อ็อบเจกต์รู้ขีดความสามารถของ server เช่น การใช้ Javabean เพื่อเขียนส่วนโปรแกรม หรือ component ที่นำกลับมาใช้ใหม่ได้

(4) late binding: เพราะว่าเอเจนท์เดินทางโดยไม่รู้ว่า server ที่จะเข้าถึงนั้นเป็น server ประเภทใด เอเจนท์เคลื่อนที่จึงจำเป็นต้องใช้ late binding ในการรวมข้อมูลของอ็อบเจกต์ขณะทำการประมวลผล

(5) การแทนความรู้ร่วม (common knowledge representation): เพราะว่าเอเจนท์ทำงานด้วยภาวะอิสระให้สำเร็จได้ด้วยตัวเอง โดยนำเอาโค้ดและข้อมูลไปด้วย ดังนั้นจึงต้องมีมาตรฐานในการนำเสนอข้อมูลและความรู้ 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

(6) การกำหนดสาระสำคัญเกี่ยวกับสภาพแวดล้อมระยะไกล: นอกจากผู้พัฒนาจะเขียนโปรแกรมเอเจนท์ในส่วนของพฤติกรรมแล้ว ผู้พัฒนาควรจะนำเสนอข้อมูลทรัพยากรที่เอเจนท์ต้องการใช้ที่เครื่องระยะไกลด้วย เนื่องจากการที่จะใช้งานบนเครื่องระยะไกลนั้น เครื่องนั้นควรมีทรัพยากรต่างๆที่เอเจนท์ต้องการ

(7) adaptive program: การทำงานของเอเจนท์มักถูกจำกัดอยู่กับสภาพแวดล้อมแบบใดแบบหนึ่ง เนื่องจาก ณ ที่นั้นๆ อาจมีทรัพยากรไม่เพียงพอ วิธีแก้ที่ง่ายที่สุดก็คือ การเคลื่อนย้ายเอเจนท์ไปยังที่อื่น แต่อย่างไรก็ตาม เอเจนท์ควรมีความสามารถในการจัดการกับข้อจำกัดเหล่านั้นได้ด้วย

(8) การวางแผนเชิงกลยุทธ์ (strategy planning): ในการวางแผนสอบถามฐานข้อมูล ผู้ใช้ควรจะสามารถเลือกทางเลือกที่ดีที่สุดก่อนที่จะส่งเอเจนท์ออกไปทำงาน ในบางกรณี mobile code ก็ไม่ใช่วิธีแก้ปัญหาที่ดีที่สุด ผู้ใช้ควรพิจารณาในเรื่องของข้อมูลของงานที่จะทำและโครงแบบของเครือข่ายด้วย

(9) โครงสร้างข้อมูลร่วม (shareable data structure): แนวคิดของการทำงานของเอเจนท์ คือ การใช้ server ที่มีทรัพยากรเตรียมไว้ให้ร่วมกัน เพราะฉะนั้น ผู้ใช้จึงต้องแบ่งปันกันใช้ทรัพยากรอย่างดีที่สุด การล็อคเป็นวิธีหนึ่งที่ดี แต่บางครั้งก็มีข้อจำกัดอยู่มาก

(10) ภาวะพร้อมกัน (concurrency control): จากความต้องการในการใช้ข้อมูลร่วมกัน ทำให้ต้องควบคุมการเข้าถึงข้อมูลพร้อมๆกัน จึงควรพิจารณาถึงการจัดการเวลาในการเข้าถึงทรัพยากรของเอเจนท์ให้ใช้ได้อย่างคุ้มค่าที่สุด

7.3 งานประยุกต์ที่เหมาะในการนำเอเจนท์เคลื่อนที่ไปใช้ การพัฒนาโปรแกรมโดยใช้หลักการของเอเจนท์เคลื่อนที่นั้น เหมาะกับงานหลายประเภท เช่น [8]

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

(2) การเฝ้าสังเกต (monitoring): เราสามารถใช้เอเจนท์ในการตรวจสอบความเปลี่ยนแปลงของทรัพยากรในเครือข่าย หรือทำการรวบรวมข้อมูลต่างๆที่ทรัพยากรเหล่านั้นสร้างขึ้นมา โดยสร้างโปรแกรมให้เอเจนท์เคลื่อนที่ทำงานเป็นช่วงเวลา แล้วนำข้อมูลที่ตรวจสอบได้ในแต่ละโหนดกลับมายังระบบเอเจนท์ ตัวอย่างเช่น การใช้เอเจนท์ตรวจสอบ server ที่ให้บริการข่าว เพื่อดูว่าข่าวใดที่ผู้ใช้สนใจ หรืออาจเป็นการตรวจสอบการใช้งานของเครือข่าย จะเห็นได้ว่าลักษณะเด่นของการทำงานของเอเจนท์ในงานประเภทนี้ 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 ซึ่งเอเจนท์แต่ละตัวจะทำหน้าที่แต่ละอย่าง

แทนคน 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

7.4 ระบบเอเจนท์ที่มีในปัจจุบัน ในหัวข้อนี้จะยกตัวอย่างระบบเอเจนท์ในปัจจุบันเพียง 2-3 ตัวอย่าง อันได้แก่

(1) Telescript: แนวคิดเรื่องเอเจนท์เคลื่อนที่ถูกนำเสนอครั้งแรกโดย General Magic ซึ่งได้พัฒนาระบบเอเจนท์ที่มีชื่อว่า “Telescript” [7] Telescript เป็นระบบแรกที่นำเอเจนท์เคลื่อนที่มาใช้ สร้างขึ้นเพื่อสนับสนุนการพาณิชย์อิเล็กทรอนิกส์ แต่ไม่ประสบผลสำเร็จนัก ภาษาที่ใช้เรียกว่า “Low Telescript” เป็นการผสมผสานระหว่างภาษาเชิงวัตถุและภาษาบทคำสั่ง (scripting language) Telescript จัดเป็นเอเจนท์เคลื่อนที่ที่มีการเคลื่อนที่ในระดับ strong mobility [10] ปัจจุบันบริษัท AT&T มีการใช้งาน Telescript เวอร์ชันใหม่ที่มีชื่อว่า “Odyssey” ซึ่งเขียนด้วยจาวา

(2) IBM aglets: IBM พัฒนาระบบเอเจนท์ชื่อว่า Aglet เขียนด้วยจาวา Aglet เป็น Java object ที่สามารถเคลื่อนที่จาก host หนึ่งไปอีก host หนึ่งในเครือข่ายอินเทอร์เน็ตได้ [9] โดยมีการใช้ graphical environment เพื่อตรวจสอบการทำงานของเอเจนท์ โปรแกรมเมอร์เพียงแต่เขียนส่วนที่เป็นคลาสขึ้นมาใหม่ก็พอ [7]

(3) TCL-Agent: เป็นระบบเอเจนท์เคลื่อนที่ที่สร้างขึ้นเพื่อการศึกษาโดย University of Dartmouth TCL-Agent ทำงานบนระบบปฏิบัติการยูนิกซ์ โดยมีตัวแปรคำสั่งของตัวเอง ซึ่งสนับสนุน strong mobility [10] จุดเด่นของ TCL-Agent คือ มีความปลอดภัยของข้อมูลสูง เพราะทำงานในสภาพแวดล้อมที่เรียกว่า Safe-TCL [7] แต่ในขณะเดียวกันก็ขาดความยืดหยุ่นในด้านของภาษา

8. ประเด็นสำคัญที่ควรพิจารณา

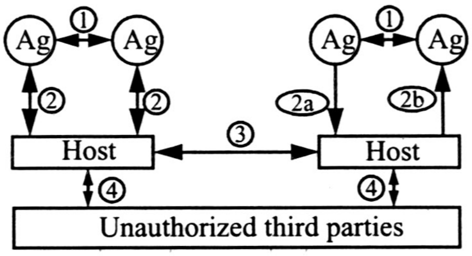
จากที่กล่าวมา ดูเหมือนว่าเอเจนท์เคลื่อนที่จะสามารถทำงานทุกอย่างตามต้องการได้ แต่การนำแนวคิดดังกล่าวไปใช้ ยังมีสิ่งที่ต้องคำนึงถึง ดังนี้

8.1 ประสิทธิภาพ (performance) จากหัวข้อที่ผ่านมาจะพบว่าเอเจนท์สามารถช่วยปรับปรุงประสิทธิภาพของระบบให้ดีขึ้น (เช่น เพิ่มแบนด์วิดท์) อาทิเช่น การนำเอเจนท์เคลื่อนที่มาช่วยในการทำ optimization ในงานทางด้านฐานข้อมูล แต่ในขณะเดียวกัน ก็ทำให้เกิดค่าใช้จ่ายสูงเช่นกัน สามารถศึกษารายละเอียดเพิ่มเติมได้จาก cost model ของ Arunaud Sahuguet [7]

8.2 ความปลอดภัย (security) จากรูปที่ 4 จะพบว่าการรักษาความปลอดภัยในระบบเอเจนท์เคลื่อนที่นั้น แบ่งเป็น 4 ส่วนด้วยกัน [5] ในส่วนที่ 1, 3 และ 4 สามารถใช้กลไกการรักษาความปลอดภัยที่มีการใช้กันอยู่ในระบบ client/server ป้องกันได้ แต่ในส่วนที่ 2 เป็นการรักษาความปลอดภัยระหว่างเอเจนท์กับ host ซึ่งต้องใช้วิธีที่เฉพาะ ส่วนที่จะต้องรักษาความปลอดภัยในส่วนนี้แบ่งออกเป็น 2 ส่วนย่อยด้วยกัน ได้แก่

(1) การป้องกัน host จากเอเจนท์: เปรียบเสมือนการป้องกันไม่ให้โปรแกรมหนึ่งๆเข้ามาทำงานใน host โดยที่โปรแกรมในที่นี้ก็คือ เอเจนท์ ซึ่งก็คือส่วนของผู้ที่ไม่ได้รับ

อนุญาต (unauthorized party) จากรูปที่ 4 นั่นเอง การป้องกัน host จากผู้ที่ไม่ได้รับอนุญาตนั้นจะป้องกันในเรื่องของความต้องการข้อมูล การแก้ไขข้อมูล การลักลอบใช้ทรัพยากรของ host และการเข้ามาแล้วทำให้ข้อมูลที่มีอยู่เสีย (เช่น ไวรัส) [9] วิธีที่ผู้ที่ไม่ได้รับอนุญาตเหล่านี้ใช้ อาจจะเป็นการลอบฟัง การปลอมตัว หรือการจู่โจมแบบไวรัส



รูปที่ 4 การรักษาความปลอดภัยในระบบเอเจนท์เคลื่อนที่

แนวคิดในการป้องกัน ก็คือ การรับรองเอเจนท์ และให้เอเจนท์ทำงานในสภาพแวดล้อมที่มีการป้องกันไว้ [5] เช่น แบบจำลอง sandbox security ซึ่งจะมีหน่วยหนึ่งของ host ที่ตัดสินใจว่าโปรแกรมใดจะสามารถทำงาน หรือโปรแกรมใดไม่สมควรที่จะทำงานบน host นี้บ้าง หรืออาจจะเป็นการใช้การเข้ารหัส ลายเซ็นอิเล็กทรอนิกส์ และ trust theory [9] เป็นต้น

(2) การป้องกันเอเจนท์จาก host: ถ้าเอเจนท์ไม่มีระบบรักษาความปลอดภัยแล้ว host อาจจะแอบดูโค้ดและ/หรือข้อมูลของเอเจนท์ได้ หรือสามารถแก้ไขสถานะและข้อมูลของเอเจนท์ได้ แต่เนื่องจากโค้ดและข้อมูลนั้นเป็นแค่ plain text และการทำงานของเอเจนท์ที่จะต้องเข้าไปทำงานใน host ทำให้การป้องกันทำได้ยาก [5]

แนวทางที่ใช้ในการรักษาความปลอดภัยได้แก่ [5]

- organizational measure: ทำการแบ่ง host ออกเป็น host ที่ไว้ใจได้ และ host ที่ไว้ใจไม่ได้ แล้วให้เอเจนท์เดินทางไปทำงานเฉพาะกับ host ที่ไว้ใจได้เท่านั้น ข้อเสียของวิธีนี้คือ การที่เราจะหาเกณฑ์ที่ใช้กำหนดว่า host นี้เชื่อใจได้หรือไม่นั้นจะขึ้นกับการกำหนดของแต่ละบุคคล นอกจากนี้การดูแลรายชื่อของ host ที่เชื่อใจได้ก็กลายเป็นระบบเปิดขนาดใหญ่
- communicating the reputation of hosts: แนวคิดนี้มีสมมติฐานว่า host ที่เอเจนท์จะเข้าไปทำงานจะเป็น host ที่เชื่อถือได้ ส่วน host ที่มีชื่อเสียงไม่ดี เอเจนท์ก็จะไม่เข้าไป
- recording the trace: การบันทึกเส้นทางการเดินทางของเอเจนท์ ข้อมูลการเดินทางจะถูกนำมาวิเคราะห์เพื่อตรวจสอบว่ามีการแก้ไขโค้ดและข้อมูลของเอเจนท์หรือไม่ แต่วิธีนี้ไม่สามารถตรวจจับการลักลอบอ่านข้อมูลได้
- blackbox: แนวคิดนี้เป็นการสร้างเอเจนท์จากเอเจนท์ที่มีอยู่ขึ้นมาใหม่ โดยใช้ conversion mechanism เปลี่ยนเอเจนท์ที่มีอยู่เป็น executable agent เรียกเอเจนท์นี้ว่าเป็นเอเจนท์ที่มีคุณสมบัติ blackbox คุณสมบัติของเอเจนท์แบบนี้ก็คือ มีการป้องกันโค้ดและข้อมูลไม่ให้สามารถอ่านหรือแก้ไขได้ คุณสมบัติ blackbox นี้จะมีช่วงเวลาที่กำหนดให้ใช้ได้ไว้ด้วย เมื่อเอเจนท์หมดอายุการใช้งาน ก็จะทำให้ข้อมูลทุกอย่างที่เอเจนท์นำมาด้วยไม่สามารถใช้ต่อได้ [11]

จากที่กล่าวมาข้างต้น จะพบว่าการรักษาความปลอดภัยของ host สามารถกระทำด้วยวิธีที่ใช้กันอยู่ในปัจจุบัน ซึ่งในเรื่องของการรักษาความปลอดภัยของเอเจนท์เคลื่อนที่นี้ ยังคงเป็นเรื่องที่ต้องศึกษาคิดค้นกันต่อไป

8.3 โครงสร้างควบคุม (control structure) ระบบเอเจนท์เคลื่อนที่ทำงานโดยการสร้างและจำลองเอเจนท์ย่อยๆ ขึ้นมา แต่การจบการทำงานของเอเจนท์ย่อยๆเหล่านั้น เป็นสิ่งที่ต้องให้ความสนใจด้วยเช่นกัน กล่าวคือ เมื่อผู้ใช้สร้างงานขึ้นมางานหนึ่ง และมีเอเจนท์กลุ่มหนึ่งถูกสร้างเพื่อทำงานนี้ให้สำเร็จ เมื่อใดก็ตามที่ผู้ใช้ต้องการจบการทำงาน เอเจนท์ทุกตัวที่ถูกสร้างขึ้นก็ควรจะจบการทำงานด้วย เราเรียกการจัด

การการทำงานในลักษณะนี้ว่า orphan management [5] เช่น search agent ตัวหนึ่งอาจทำการปล่อย search agent ออกมาจำนวนหนึ่ง เพื่อค้นหาข้อมูลแบบขนาน ด้วยการจัดการที่เรียกว่า orphan management นี้ กระบวนการที่เกิดขึ้นมาใหม่จะไม่ขึ้นกับกระบวนการที่สร้างมัน แต่จะขึ้นกับเอเจนท์อีกตัวที่มีหน้าที่ในการเก็บรวบรวมผลลัพธ์จากการค้นหา นั่นคือ ต้องมีการเลิก (terminate) มี orphan detection protocol และมีการกำหนดความสัมพันธ์ที่ขึ้นแก่กันของเอเจนท์

8.4 การสนับสนุนการทำรายการเปลี่ยนแปลง (transactional support) ในปัจจุบัน โปรแกรมประยุกต์ที่สร้างขึ้นเพื่อการพาณิชย์นั้น ต้องมีความสามารถในการทำรายการได้สูง ดังนั้น เมื่อแนวคิดของเอเจนท์ถูกนำมาใช้ในการสร้างโปรแกรมประยุกต์ทางพาณิชย์ การใช้งานโปรแกรมประยุกต์ที่ใช้เอเจนท์ในการจัดการรายการ จะต้องบรรลุวัตถุประสงค์ 2 ข้อด้วยกัน คือ

(1) แบบจำลองรายการต้องสามารถทำงานที่ต่อเนื่องเป็นเวลานานได้ เพื่อให้รายการนั้นทำงานได้สำเร็จลุล่วง เอเจนท์สามารถตอบสนองความต้องการนี้ได้ เนื่องจากเอเจนท์มีการทำงานแบบไม่ประสานเวลา ดังนั้น การทำงานนานๆ แบบการทำรายการนั้นไม่ใช่ปัญหาของเอเจนท์

(2) การทำรายการหนึ่งๆ หากเกิดข้อผิดพลาดขึ้นมาจะต้องสามารถนำข้อมูลต่างๆกลับมาได้ เช่น เมื่อเอเจนท์ที่ทำหน้าที่ซื้อตั๋วหนัง ทำการโต้ตอบกับ ticket server เพื่อซื้อตั๋วหนัง ระหว่างการโต้ตอบนั้น เอเจนท์ได้รับตั๋วอิเล็กทรอนิกส์ ในขณะที่ ticket server ก็ได้รับเงินสดอิเล็กทรอนิกส์ (การโต้ตอบนี้ได้ทำการเปลี่ยนแปลงสถานะของ server และของเอเจนท์) แต่ถ้ามีการยกเลิกรายการดังกล่าว ไม่ว่ากรณีใดๆก็ตามจะต้องคืนค่าสถานะทั้งสองได้อย่างถูกต้อง

ในปัจจุบัน แนวทางที่ใช้ในการรวม mobile program และการทำรายการเข้าด้วยกัน คือการใช้ Java Database Connectivity (JDBC) และ Java Transaction Service (JTS) [5]

9. สรุป

แม้ว่าการนำเอเจนท์เคลื่อนที่ไปใช้ยังไม่เป็นที่แพร่หลายมากนัก แต่ด้วยข้อดีในเรื่องของการกระจายซอฟท์แวร์ตามความต้องการ การลดค่าใช้จ่ายในการสื่อสาร การทำงานแบบไม่ประสานเวลา และความสามารถในการเคลื่อนที่ได้ จึงทำให้เอเจนท์เคลื่อนที่เป็นที่สนใจกันอย่างมาก ทั้งนี้ จะต้องคำนึงถึงเรื่องความปลอดภัยของข้อมูล การควบคุมการทำงานของเอเจนท์ และการทำงานของเอเจนท์ที่สนับสนุนการทำรายการ รวมทั้งการลงทุนในการสร้างระบบเอเจนท์เคลื่อนที่ที่สูง ไม่ว่าจะเป็นการให้บริการซีพียู หน่วยความจำ และสภาพแวดล้อมต่างๆ

เอกสารอ้างอิง

[1] Friedemann, M. and Kotz, D. 2000. **Agent System, Mobile Agents, and Application.** [Online]. Available: http://www.inf.ethz.ch/department/IS/vs/publ/papers/ ASA-MA-abstract.html.

[2] Dale, J. 1995. “A Mobile Agent Architecture to Support Distributed Resource Information Management.” Faculty of Engineering and Computer Science, University of Southampton.

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

[4] Torsten, S. 1996. **Execution Environments for Mobile Code- A Cornerstone for Adaptable Distributed Systems.** (German) Research Report, University of Kaiserslautern. [Online]. Available: http://www.newcastle.research.ec.org/cabernet/research/radicals/1996/delegates/stolpmann.html.

[5] Rothermel, K. and Schwehm, M. 1998. **Encyclopedia of Computer Science and Technology.** New York.

[6] Jim, W. 1996. **Mobile Agents White Paper**. [Online]. Available: http://www.iiuf.unifr.ch/~chantem/white_whitepaper/index.htm.

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

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

[9] Hurst, L. 1996. **TNET Executive Summary**. [Online]. Available: http://www.cs.tcd.ie/research_groups/aig/iag/pubreview/chap5/chap5.html.

[10] Fuggetta, A., Picco, G. pietro, and Vigna, G. 1998. **Understanding Code Mobility**. [Online]. Available: http://citeseer.nj.nec.com/fuggetta98understanding.html.

[11] Hohl, F. 1998. **Time Limited Blackbox Security: Protecting Mobile Agent From Malicious Host.**