CÔNG NGHỆ ĐA TÁC TỬ DI ĐỘNG VÀ ỨNG DỤNG ĐỂ THƯƠNG LƯỢNG TRONG THƯƠNG MẠI ĐIỆN TỬ

"MOBILE AGENT TECHNOLOGY AND APPLICATION IN NEGOTIATION IN E-COMMERCE"

PGS.TS Đoàn Văn Ban[(1)], CN Quách Xuân Trưởng[(2)]

(1) Viện Công nghệ thông tin - Viện KH&CN Việt Nam
(2) Khoa Công nghệ thông tin - Đại học Thái nguyên

Abstract: Mobile agent is the model where agents have the properties of autonomy and mobility in traveling from one host to another in order to complete tasks. Primarily, mobile agent aims at transferring execution nearby data sources, which help to reduce network load, overcome network latency, support asynchronous execution, and create strong adaptability in unhomogeneous environments. With these advantages, mobile agent commits a new, effective, and easy-to-use method in developing network applications, especially distributed applications. This report is about the mobile agent model, some mobile agent development tools, and one application of mobile agent in e-commercial negotiation which is one of online service attracting attentions in the process of Vietnam economic integration in particular.

1. Giới thiệu

Cùng với sự phát triển của các kỹ thuật tiên tiến về máy tính, kỹ thuật truyền tin và các lĩnh vực tin học hiện đại (như hệ thống phân tán, AI & Khoa học nhận dạng, máy học, mã di động, truy tìm thông tin, cơ sở dữ liệu và cơ sở tri thức ..). Kết hợp với phương pháp lập trình hướng đối tượng đã tạo ra một phương pháp phát triển mới. Phương pháp lập trình hướng tác tử.

Tác tử di động là một thành phần phần mềm có khả năng di chuyển một cách tự trị từ nút mạng này sang nút mạng khác và thực hiện các xử lý thay thế cho con nguời để thực hiện các công việc được giao. Khi di chuyển, các tác tử di động đóng gói mã nguồn, dữ liệu và có thể cả trạng thái thi hành. Như vậy, tác tử di động có thể dừng việc thi hành đang thực hiện tại máy này, di chuyển sang máy khác và khôi phục lại sự thi hành tại máy đích.

1.1. Một số tính chất đặc trưng của tác tử di động

- ***Tính di động***: Là khả năng di chuyển từ môi trường thi hành này sang môi trường khác của một tác tử. Khả năng di động của một tác tử được phân thành hai loại. Di động mạnh là khả năng mà hệ thống có thể di chuyển cả mã chương trình và trạng thái thi hành của tác tử đến một môi trường khác. Di động yếu là khả năng của hệ thống chỉ có thể di chuyển mã chương trình giữa các môi trường thi hành với nhau, mã nguồn có thể mang kèm theo một số dữ liệu khởi tạo nhưng trạng thái thi hành thì không thể di chuyển.
- ***Tính tự trị***: Là khả năng tự kiểm soát những hoạt động của chính nó và thi hành các tác vụ độc lập với người dùng hoặc của tác tử khác. Có nhiều hướng đánh giá về sự tự trị của tác tử, trong đó hai đặc tính: hướng đích và chủ động thường được dùng để đánh giá mức độ tự trị

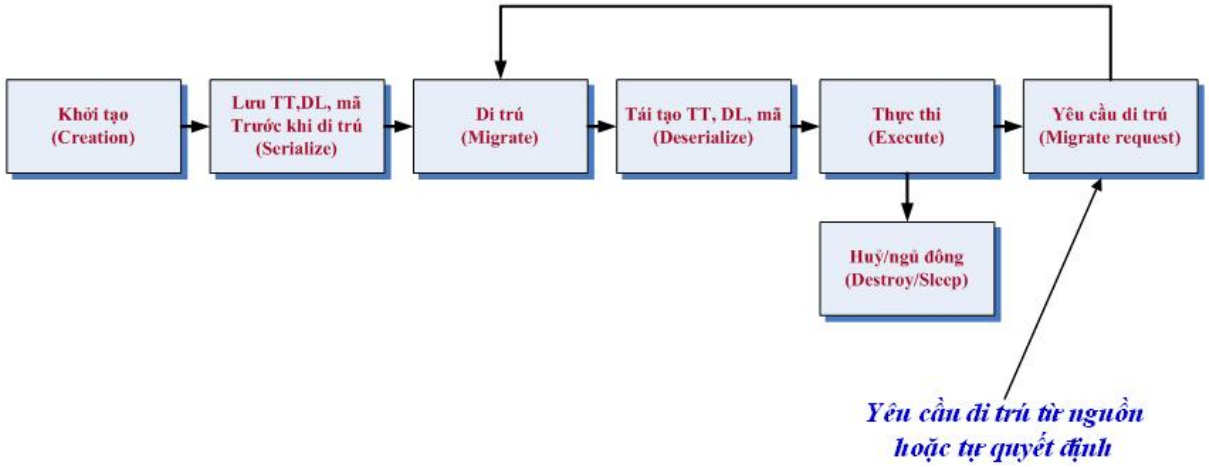
của tác tử. Khả năng tự trị của tác tử chủ yếu được quyết định bởi tri thức trang bị cho tác tử [9].

- ***Tính thích nghi*** : Là khả năng của tác tử có thể thực thi trên những môi trường lạ và biết cách học và thay đổi hành vi của nó theo các tri thức thu được từ môi trường.
- ***Khả năng cộng tác***: Là khả năng liên lạc, phối hợp hoạt động của các tác tử với các tác tử khác của cùng môi trường hay với các loại đối tượng trong những môi trường khác.
- ***Tính đóng và bền vững***: Là độc lập với các phần mềm tác tử khác và có khả năng tồn tại bền vững trong môi trường hoạt động.

1.2. Nguyên lý hoạt động

1.2.1. Vòng đời của một tác tử di động

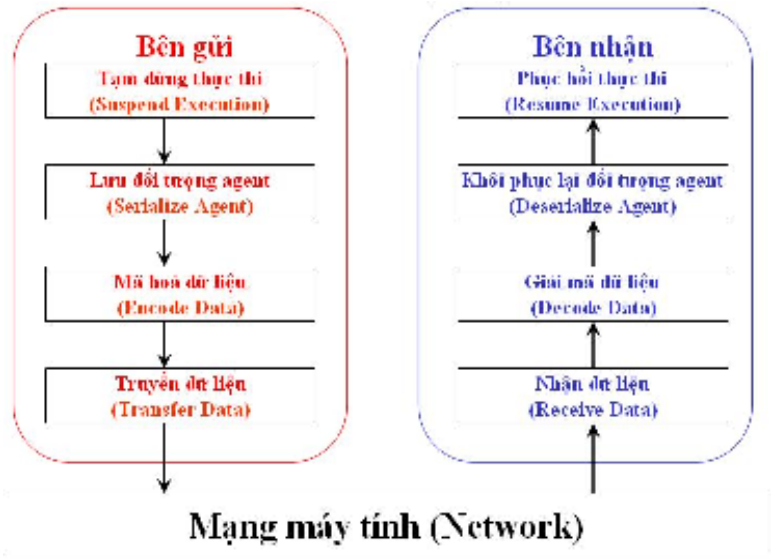
Mọi tác tử di động đều thực hiện tuần tự theo logic: Đầu tiên, tác tử được tạo ra, di trú từ host này sang host khác theo lịch trình, thực hiện các nhiệm vụ được giao và cuối cùng bị huỷ sau khi đã hoàn thành nhiệm vụ.



Hình 1.1. *Vòng đời của một tác tử di động.*

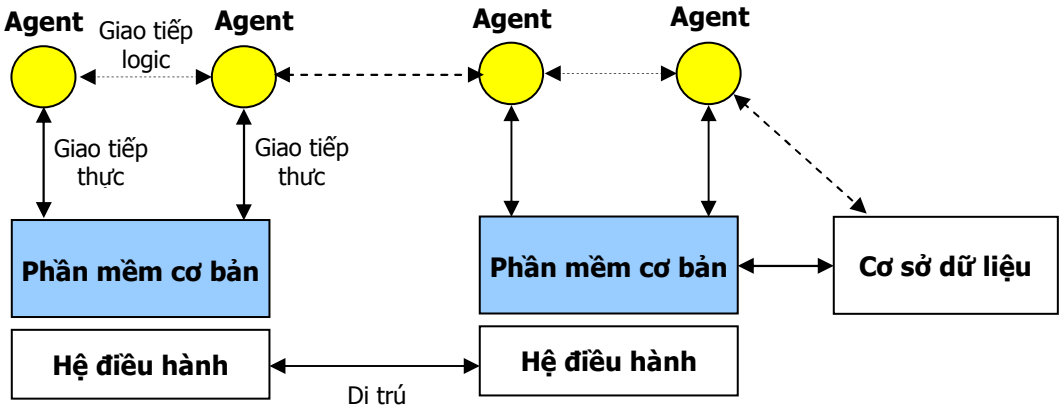
1.2.2. Cơ chế di chuyển của tác tử di động.

Tác tử sẽ tự quyết định đóng gói và di trú đến môi trường khác để thực thi là đặc điểm nổi bật của tác tử di động. Khi có nhu cầu di trú tác tử sẽ tạm dừng thực thi mã, thực hiện lưu trạng thái của đối tượng (có thể có khả năng lưu lại cả trạng thái thi hành). Sau khi mã hoá dữ liệu tác tử sẽ được truyền đi trên hệ thống mạng. Khi đến đích, tác tử sẽ thực hiện các bước ngược lại với bên gửi để khôi phục lại trạng thái của tác tử và tiếp tục thực hiện nhiệm vụ.



Hình 1.2. *Cơ chế di chuyển của mobile agent*

1.3. Kiến trúc của hệ thống tác tử di động.



Hình 1.3. *Kiến trúc của một hệ thống tác tử di động*

Hệ điều hành của các hệ thống máy tính cung cấp các phần mềm cơ bản để thực hiện việc tích hợp các tác tử di động vào hệ thống và cung cấp đầy đủ các chức năng cơ bản cho tác tử di động hoạt động. Phần mềm cơ bản là giao diện giữa tác tử di động và hệ điều hành của máy tính [11].

Một tác tử di động có thể giao tiếp trực tiếp với các phần mềm cơ bản bằng cách yêu cầu một thủ tục mà phần mềm cơ bản cung cấp và nhận câu trả lời bằng các tham số trả về. Các tác tử đang cùng hoạt động tại một hệ thống cũng có thể giao tiếp với nhau. Việc truyền thông này có thể được thực hiện theo hai cách:

1. *Cách thứ nhất: giao tiếp thực* - hai tác tử giao tiếp trực tiếp với nhau bằng cách gửi cho nhau các thông điệp hoặc yêu cầu các thủ tục. Cách này linh hoạt và cho phép tác tử mức độ tự do

cao. Tuy nhiên, phần mềm cơ bản (cũng là các server) khó theo dõi và điều khiển các hoạt động của các tác tử.

2. *Cách thứ hai: giao tiếp logic* - hai tác tử giao tiếp với nhau thông qua phần mềm cơ bản. Các tác tử chỉ giao tiếp trực tiếp với phần mềm cơ bản. Cách giao tiếp giữa hai tác tử này bao gồm hai kết nối thực giữa các tác tử cần giao tiếp với phần mềm cơ bản. Cách này hạn chế hơn và dễ bị thay đổi hơn.

Nếu một tác tử muốn truy xuất thông tin từ một cơ sở dữ liệu bên ngoài thì tác tử đó phải thông qua phần mềm cơ bản. Phần mềm cơ bản sẽ truy xuất đến cơ sở dữ liệu và thực hiện các công việc mà tác tử yêu cầu (như tìm kiếm,…) sau đó trả kết quả về cho tác tử.

Phần mềm cơ bản của các hệ thống tác tử di động gồm có ba tầng: tầng tác tử, tầng an ninh, tầng truyền thông:

- Tầng tác tử cung cấp các tác vụ chính cho việc thi hành và kiểm tra của tất cả các tác tử trên máy. Ngoài ra, nó cung cấp cho tất cả các tác tử môi trường làm việc và sự thi hành các tác tử độc lập với nhau. Tầng tác tử còn cung cấp các chức năng cơ bản cho hoạt động của các tác tử.
- Tầng an ninh cung cấp các chức năng cho phép truyền các thông điệp và các đối tượng trên mạng một cách an toàn.
- Tầng truyền thông bao gồm các đặc tả cho các giao thức truyền, các định dạng tài liệu, đối tượng.

1.4. Chuẩn hoá

Các hệ tác tử di động khác nhau có thể khác nhau về kiến trúc và sự thực thi điều này gây cản trở không ít cho khả năng cùng vận hành giữa các phần và sự triển khai nhanh chóng của công nghệ tác tử di động trên thị trường. Trước nhu cầu đó, OMG đã phát triển chuẩn MASIF (Mobile agent System Interoperability Facility) - cho phép các tác tử cùng vận hành [20]. MASIF đưa ra các giao diện giữa các hệ thống tác tử, giữa các ứng dụng tác tử và các hệ thống tác tử. Trong đó, giao diện giữa các hệ thống tác tử thích hợp cho các nhà phát triển ứng dụng còn giao diện giữa các ứng dụng tác tử và hệ thống tác tử cho phép các tác tử di động di chuyển qua nhiều host trong môi trường mở. MASIF không phải là ngôn ngữ cho khả năng cùng vận hành, nó định nghĩa các giao diện ở mức hệ thống tác tử hơn ở mức tác tử.

Bên cạnh MASIF, FIPA (Foundation for Intelligent Physical Agent) là tổ chức các chuẩn cho các tác tử vật chất có tính thông minh. FIPA đẩy mạnh công nghệ dựa trên tác tử và khả năng cùng vận hành của các chuẩn của nó với các công nghệ khác [5]. FIPA được thành lập năm 1996 với mục đích đưa ra các đặc tả cho các hệ thống tác tử và dựa trên tác tử không thuần nhất và tương tác lẫn nhau. FIPA đóng vai trò chủ yếu trong sự phát triển của các chuẩn của các tác tử và đẩy mạnh các sáng kiến

và các sự kiện góp phần vào sự phát triển của công nghệ tác tử. Hơn nữa nhiều ý tưởng được bắt đầu và được phát triển trong FIPA đang hình thành tiêu điểm mới trong thế hệ mới của công nghệ Web/Internet và các đặc tả có liên quan.

FIPA, tổ chức các chuẩn cho các hệ thống tác tử và đa tác tử được IEEE (Institute of Electrical and Electronic Engineers) chấp nhận chính thức vào ngày 08/06/2005. Hiện nay, các chuẩn cho các hệ thống tác tử và dựa trên tác tử đang được đưa vào ngữ cảnh lớn hơn của sự phát triển phần mềm, công nghệ tác tử cần được làm việc và tích hợp với các công nghệ khác [28].

FIPA và MASIF hợp tác lẫn nhau trong đó FIPA tập trung vào tính thông minh còn MASIF thì lại tập trung vào tính di động của tác tử.

2. Công cụ phát triển tác tử di động.

Hiện nay có nhiều hệ thống tác tử di độngs đã được các công ty, tổ chức danh tiếng xây dựng như Aglets của IBM, Voyager của Object Space, Mole của Đại học Stuttgart (Đức), Telescript của General Magic, JADE của Telecom Italia Lab, Concordia của Mitsubishi Electric Lab, … mỗi hệ thống đều được sử dụng cho các mục đích riêng, trong các lĩnh vực ứng dụng khác nhau. Đa số các hệ tác tử di độngs sử dụng Java để hỗ trợ phát triển ứng dụng nhưng mỗi hệ thống có đặc thù riêng của nó.

2.1. Java – ngôn ngữ hiệu quả dùng để phát triển tác tử

Có nhiều yêu cầu về công nghệ để cài đặt các hệ tác tử di động. Cơ bản bao gồm các yêu cầu về cấu trúc của phương tiện tính toán, môi trường mà các tác tử đó hoạt động. Các host phải được thiết kế, cài đặt và triển khai không chỉ cho phép các tác tử thực hiện mà còn phải thực hiện một cách an toàn. Hiện nay, chưa có một hệ thống nào có thể đáp ứng được hoàn toàn các yêu cầu đề ra cho một hệ tác tử di độngs [13]. Java do Sun Microsystem phát triển được xem là môi trường tốt nhất hiện nay chứa những đặc tính cần thiết cho việc xây dựng các hệ thống tác tử di độngs [5] [13]. Java là một ngôn ngữ lập trình hướng đối tượng cho mạng hay được gọi là ngôn ngữ của Internet [2] [5] [13]. Một số đặc điểm (như độc lập với môi trường, nạp lớp động, lập trình đa luồng, hỗ trợ đối tượng phân tán, tuần tự hoá đối tượng, thi hành an toàn, …) cho thấy Java là một ngôn ngữ hiệu quả dùng cho lập trình các tác tử.

2.2. Công cụ phát triển Aglets Workbench.

Aglets Workbench được xây dựng và phát triển bởi D.B.Lange và IBM Tokyo Research Laboratory. Aglets Workbench bao gồm: bộ công cụ phát triển aglets (ASDK) và platform để các aglet thi hành trên đó. Hiện nay bộ Aglets Software Development Kit (ASDK) do IBM phát triển đã dừng lại ở phiên bản 1.1beta3 trên nền JDK1.1. Phiên bản mới nhất của ASDK là 2.0.2 do SourceForge phát triển trên nền JDK1.3

- *Các thành phần cơ bản của Aglets*

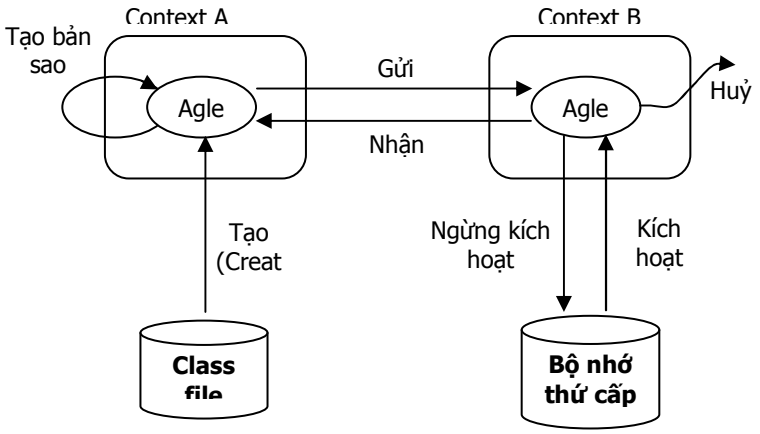
- Aglet: là một đối tượng Java, có khả năng di chuyển đến các host trên mạng.

- Proxy: Là đối tượng đại diện cho Aglet. Có nhiệm vụ bảo vệ Aglet tránh khỏi các truy cập trực tiếp đến các phương thức public. Ngoài ra, nó có thể ẩn đi vị trí thực của Aglet.
- Context: Là workplace của Aglet, cung cấp các dịch vụ để duy trì và quản lý các hoạt động của các Aglet. Bảo vệ host tránh khỏi các aglet độc hại.
- Message: Là đối tượng dùng để trao đổi thông tin giữa các aglet. Aglet trao đổi thông tin với nhau bằng cách truyền các thông điệp. các thông điệp này có thể được truyền đồng bộ hoặc không đồng bộ.

- *Các hoạt động cơ bản của Aglet*

Lớp ibm.com.aglet.Aglet cung cấp các chức năng cơ bản cho một đối tượng mobile và mọi aglet phải là thể hiện hoặc lớp con của nó. Để sử dụng một aglet, trước hết phải tạo ra thể hiện (Instantiated) của nó. Có hai cách để làm điều này:

Cách 1: tạo một aglet hoàn toàn mới từ định nghĩa lớp bằng cách gọi hàm AgletContext.createAglet(URL Codebase, String name, Object init). Hàm này tạo ra một thể hiện mới trong context và khởi động nó nếu cần thiết.



Hình 3.2. *Mô hình vòng đời và hoạt động*

Cách 2: Tạo ra một bản sao của một aglet đã tồn tại bằng cách dùng hàm Aglet.Clone(). Aglets được sao chép ra có cùng trạng thái như aglets gốc nhưng có aglets ID khác.

Khi được tạo ra, một đối tượng aglet có thể được gửi đi đến và/hoặc nhận về từ một server khác, ngưng hoạt động và được lưu trữ để rồi lại được tái kích hoạt.

Một aglet có thể tự gửi chính nó đến một server khác bằng cách gọi hàm nguyên gốc Aglet.dispatch(URL dest). Aglet lưu trú trong một context và có thể di chuyển từ context này đến context khác trong suốt quá trình hoạt động. Do server có thể phục vụ nhiều context trong cùng một máy ảo Java (JVM). Và một host có thể phục vụ nhiều server nên context được đặt tên với những thuộc tính sau:

- Địa chỉ của host, thường là địa chỉ IP.
- Port mà server dùng để nghe.
- Tên context trong server

Ví dụ: atp://aglets.ibm.com:1434/context_name

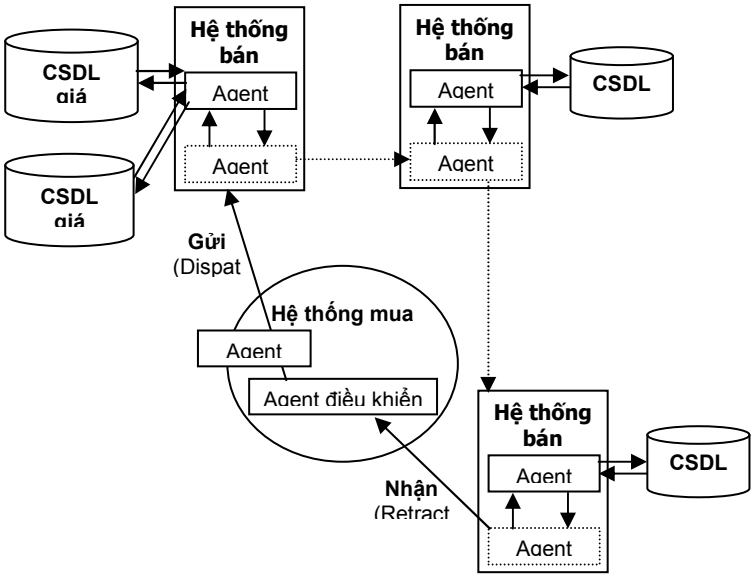
Aglet có bộ thư viện các hàm API, bao gồm các hàm chuyên biệt dành cho việc phát triển các tác tử. Khi aglet được tạo ra, nó sẽ chạy trên mọi máy có hỗ trợ aglet API mà không quan tâm đến nguồn gốc của hệ điều hành và phần cứng bên dưới.

3. Ứng dụng tác tử di động trong thương mại điện tử

Công nghệ tác tử di động rất thích hợp cho sự phát triển của thương mại điện tử. Nó phù hợp cho các ứng dụng đòi hỏi một số lượng lớn các tương tác đóng và truyền thông trên mạng. Một tương tác thương mại điện tử đòi hỏi nhiều tương tác giữa client site (người bán) và server site (người mua). Vì vậy, mô hình tác tử di độngs cung cấp một mô hình thích hợp cho thương mại điện tử trên Internet. Để làm được điều đó, các tác tử sẽ tự động hoá một phần hay toàn bộ các hoạt động kinh doanh trong thương mại điện tử. Trong đó, các tác tử có thể đóng vai trò là người mua, người bán, người môi giới, người cung cấp thông tin,… để thực hiện các giao dịch.

3.1. Mô hình thương mại điện tử sử dụng tác tử di động để thương lượng.

Dựa trên mô hình tác tử di độngs, một hệ thống thương mại điện tử cơ bản bao gồm hai module chính: Hệ thống bán và Hệ thống mua. Các tác tử bán đưa ra các sản phẩm và dịch vụ của nhà cung cấp. Các tác tử này tĩnh và nằm ở bên trong site của nhà cung cấp.Các tác tử mua thu thập thông tin từ các tác tử bán. Chúng có khả năng di động để di chuyển từ site của nhà cung cấp này sang site của nhà cung cấp khác. Các tác tử mua có khả năng giao tiếp với các tác tử bán bằng các truy vấn SQL đến CSDL thông qua giao diện JDBC.

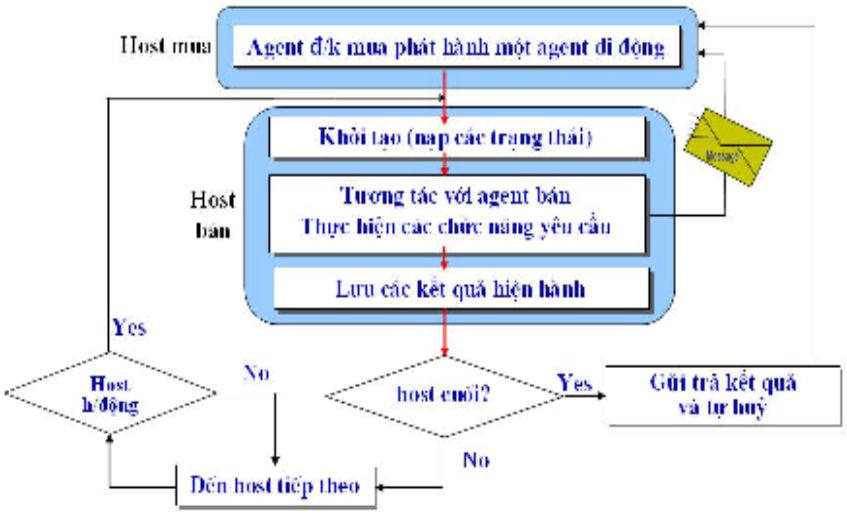


Hình 3.1. Mô hình TMĐT sử dụng mobile agents.

3.2. Xây dựng ứng dụng minh hoạ tác tử thương lượng

Ví dụ được xây dựng bằng ngôn ngữ Java dựa trên nền tảng Aglet Workbench mô phỏng theo mô hình trên (hình 3.1).

Hệ thống bán bao gồm hai lớp GuiShop.class và DataProcessing.class. Lớp GuiShop là một tác tử tĩnh hoạt động tại host bán, làm nhiệm vụ giao tiếp với các tác tử từ nơi khác đến. Lớp DataProcessing chứa các phương thức cho phép tác tử bán tương tác với cơ sở dữ liệu.



Hình 3.2. *Sơ đồ hoạt động của hệ thống*

Hệ thống mua bao gồm các lớp: Abstract Itinerary: Định nghĩa một mô hình lịch trình của tác tử với hai phương thức trừu tượng là go() và hasMoreDestinations(). Lớp Itinerary cài đặt giao diện Java.io.Serializable. Lớp Serializable giúp cho việc quản lý đối tượng mà muốn lưu trạng thái của đối tượng để sử dụng lại.

- SepItinerary: Kế thừa và cài đặt các phương thức của lớp trừu tượng Itinerary và lưu giữ dấu vết tại đích hiện tại của tác tử di động. Phương thức go() được khai báo chồng (overloaded), go(URL) sẽ được gọi để tác tử gửi (dispatch) nó đến đích tiếp theo trong lịc trình. Phương thức go() sử dụng tính chất đệ quy để điều khiển cuộc hành trình, cho phép tác tử bỏ qua các host không có hiệu lực trong lịch trình. Các host không có hiệu lực sẽ là nguyên nhân để phương thức dispatch được gọi trong phương thức go(URL) ném ra một ngoại lệ, Nó sẽ bỏ qua và gọi lại phương thức go().

- Abstract Childrens: là một lớp trừu tượng cho phép kế thừa để cài đặt các lớp tác tử di động cụ thể.

- BuyerChild và GetListChild là hai lớp kế thừa lớp trừu tượng Childrens cài đặt hai tác tử di động làm nhiệm vụ thương lượng giá sản phẩm và tìm kiếm thông tin danh sách sản phẩm. GetListChild sẽ đến lần lượt các host trong lịch trình, tương tác với tác tử bán và gửi về BuyerMaster thông tin các sản phẩm có bán tại host bán. BuyerChild là tác tử thương lượng, nó đi đến các host trong lịch trình và thực hiện việc thương lượng giá sản phẩm mà BuyerMaster yêu cầu với tác tử bán, nó sẽ lưu giữ lại thông tin về giá thấp nhất và nơi bán giá thấp nhất và tiếp tục thực hiện thương lượng trên các host khác cho đến khi hết lịch trình. Kết thúc lịch trình nó sẽ gửi về BuyerMaster thông tin về giá và địa chỉ nơi bán có giá thấp nhất. Các tác tử này sẽ sử dụng phương thức go() của lớp Itinerary để tự nó di truyển đến các host trong lịch trình. Sau khi hoàn thành nhiệm vụ các tác tử này sẽ tự động huỷ.

- BuyerMaster là một tác tử hoạt động tại host mua, làm nhiệm vụ khởi tạo các tác tử di động và cung cấp cho các tác tử lịch trình và các thông tin cần thiết cho các tác tử thực thi nhiệm vụ. Sau khi các tác tử được gửi đi đến các host, thực hiện các nhiệm vụ của mình và gửi trả kết quả về BuyerMaster .

4. Kết luận

Trong bài báo này chúng tôi đã giới thiệu về mô hình tác tử di động, một hướng phát triển mới cho các ứng dụng phân tán. Nó đưa ra một phương pháp lập trình mới, hướng tác tử gần gũi hơn với cách tư duy và hoạt động của con người. Tuy còn nhiều khó khăn trong việc triển khai hệ thống tác tử di động trong thực tế, nhưng mô hình này có nhiều tiềm năng trong một số ứng dụng. Đặc biệt là ứng dụng trên mạng máy tính, trong đó các ứng dụng trong lĩnh vực thương mại điện tử được xem là rất thích hợp với mô hình này. Do đó việc nghiên cứu để gải quyết những vấn đề còn tồn tại của mô hình trên là rất cần thiết và cần được đầu tư nghiên cứu.

TÀI LIỆU THAM KHẢO

TIẾNG VIỆT

1. Đoàn Văn Ban, *Lập trình hướng đối tượng với Java,* Nhà xuất bản Khoa học và Kỹ thuật, 2003.
2. Hoàng Ngọc Giao, *Java và ứng dụng mạng*, Nhà xuất bản Thống kê, 2000.
3. Nguyễn Phương Lan, Hoàng Đức Hải, *Java lập trình mạng*, Nhà xuất bản Giáo Dục, 2001.
4. Trần Đình Quế, Nguyễn Mạnh Sơn, Nguyễn Mạnh Hùng, *Nghiên cứu phát triển kỹ thuật và kiến trúc hệ phần mềm dựa trên agent cho thương lượng tự động trong thương mại điện tử thế hệ thứ 2*, Báo cáo đề tài nghiên cứu 58-04-KHKT-RD, Tổng Công Ty Bưu Chính Viễn Thông, 2005.
5. Hà Dương Tuấn, "Ghi chú về chuẩn CORBA, chuẩn XML, ngôn ngữ Java và công nghệ tác tử", *Hội nghị hè 2000*.

TIẾNG ANH

6. Y. Aridor, Mitsuru Oshima, "Infrastructure for Mobile Agents: Requirements and Design", *Proceeding of the 2th international workshop on Mobile Agents*, Springer-Verlag, 1998.
7. R.Ashi, I.Rahwan et al, "Architectures for negotiation agents", *In V.Marik et al. Editors, MAS and Application III, Proceedings of CEEMAS2003*, Vol. 2691, Lecture Note in AI, Springer Verlag, 2003.
8. C. Bartolini, C. Preist, N.R. Jennings, *Architecting for Reuse: A Software Framework for Automated Negotiation*, in F. Giunchiglia, J. Odell, G. Weiß (Eds.) Agent-Oriented Software Engineering III, Springer-Verlag, 2003.
9. J. Bradshaw, "An Introduction to Software Agents", *In Software Agents* ed. J. Bradshaw. Menlo Park, Calif.: AAAI/ The MIT Press, 1997.

10. W. Brenner, R. Zarnekow, H. Wittig, *Intelligent Software Agent Foundations and Applications*, Springer, 1998.
11. W.R. Cockayne, M. Zyda, *Mobile agents: Explanations and Examples*, Manning, 1997.
12. L. Cogoi et al, "Seamless Access to Databases through KQML in an Agent-enriched Web", *Workshop "Dagli oggetti agli agenti: tendenze evolutive dei sistemi software"*, Parma, Maggio, 2000.
13. J. Conde, "Mobile agents in Java", CERN Particle Physics Laboratory, Technical Report CERN/IT/ASD/RD45/98/12, 1998.
14. A. Corradi, R. Montanari, C. Stefanelli, "Mobile agents Integrity in E-Commerce Applications", *Proceedings of the 1999 ICDCS Workshop on Electronic Commerce and Web-Based Applications*, Institute of Electrical and Electronics Engineers, 1999.
15. A. Fuggetta, G.P. Picco, G. Vigna, "Understanding Code Mobility", *IEEE transactions on software engineering*, Vol. 24 (5), 1998.
16. M. Greenstein, M. Vasarhelyi, *Electronic commerce: Security, Risk Management and Control,* McGraw-Hill, 2002.
17. W. Jansen, T. Karygiannis, "Mobile Agent Security", *National Institute of Standards and Technology*, Special Publication 800-19, 1999.
    http://csrc.nist.gov/mobilesecurity/Publications/sp800-19.pdf
18. N.R. Jennings, M. Woolridge, *Applications of Intelligent Agents,* Springer Verlag/ Berlin/ Heidelberg, 1998.
19. M. Knappik, J. Johnson, *Developing Intelligent Agents for Distributed Systems*, McGraw-Hill, 1998.
20. D. B. Lange, "Mobile Objects and Mobile Agents: The Future of Distributed Computing?", *In Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, 1998.
21. D. B. Lange, M. Oshima, "Seven Good Reasons for Mobile Agents". *Communications of the ACM,* 42(3): 88 – 89, 1999.
22. R. Lomuscio, M. Wooldridge, N. R. Jenning, "A Classification Scheme for Negotiation in Electronic Commerce", *International Journal of Group Decision and Negotiation*, 2003.
23. D.S. Milojicic, W. LaForge, D. Chauhan, "Mobile Objects and Agents (MOA)", *Distributed Systems Engineering*, 1998.
24. Vu Anh Pham, Ahmed Karmouch, "Mobile Software Agents: An Overview", *IEEE Communications Magazine,* July 1998: 26 – 37, 1998.
25. J.E. White, "Mobile Agents", In *Software Agents* ed. J. Bradshaw. Menlo Park, Calif.: AAAI/ The MIT Press, 1997.

26. D. Zeng, K. Sycara, "Bayesian Learning in Negotiation", *International Journal of Human Computer System*, Vol. 48, pp 125 – 141, 1998.
27. http://aglets.sourceforge.net