Controlling Access to Music Selection System on Mobile Agent Architecture

Khin Wai Moe, Naychi Lai Lai Thein
Computer University (Hinthada)
khinwaimoe@gmail.com

Abstract

This paper describes the mobile agent-based music selection that generates dynamic playlist based on suggestions from multiple users and presents the design and development of music selection service which is implemented using the mobile agent architecture. Controlling Access to Music Selection System on mobile agent architecture can generate dynamic playlist based on the user requests. Music selections are performed autonomously by agents. A music service allows users participating in selecting music.

This system consists of three types of stationary agents and one mobile agent. Vote collector (VC) mobile agent collects all votes from each voter (VO) which behalf of each user and sends candidate music lists to vote manager (VM) agent which dispatch candidate music lists from vote administrator (VA) agent to VC. Then VM forward received candidate music with votes to VA again. Finally VA generates playlist with most demand music selected from each user.

Keyword: Mobile Agent

1. Introduction

Mobile agents are created by a distributed application at a computer site and launched to another site using an underlying mobile agent platform. An instance of the platform running at the remote site can receive the mobile agent and dispatch it to the distributed application running at that site. A family of computer applications in electronic government can be suitably carried out using a combination of mobile and stationary agents, reducing network traffic by allowing the least number of interactions across platforms. An example of such application is electronic voting.

Controlling Access to Music Selection System on mobile agent architecture builds an understanding how music is currently listened to people. Playlist is created based on users' votes collected by mobile agent. Mobile vote collector VC agent which co-ordinates with a stationary vote manager would visit each of the voters site and collect their vote for music they desired. This service helps increasing the level of participatory social activities collaborative entertainment.

In this paper, section 2 discusses related areas and problem issues of the system. Session 3 explains mobile agent of the system. The system design and detailed implementation are presented in section 4 and the last is conclusion.

2. Related Areas and Problem Issues

Selection of requested music and generating playlist are done by a staff or music selector –a disk-jockey (DJ) who need always online. In Jukola, O'Hara [6] show an interactive MP3 Jukebox device designed to allow a group of people. In public DJ, Leitich [10] presented round base multiplayer game is used to submit the music tracks to a server which need connection each time music is submit. Komninos [1] presented the use of mobile agent as DJ and use TEEMA framework for agent development. Benedicenti [7] also presented the use of TEEMA framework for mobile agent development of collaborative media sharing. The Party Vote system [2] takes the voting concept from Jukola and applies it in small group situations such as parties.

Some issues related to mobile agent are as follows:

- Execution performance: often sacrificed in order to achieve portability via code interpretation. Justin time compilation is one way towards better performance.
- Robustness: involves fault tolerance schemes (e.g., coping with hosts crashing) and is key for trust and acceptance of the technology in companies.
- Location transparency: requires a mechanism to keep track of the places where the agents are executing at every moment.
- Stability mobile agents can be sent when the network connection is alive and return results when the connection is re-established.

- Trusted and un-trusted: Some mobile agent platforms treat trusted and un-trusted agent different. Un-trusted agents are prohibited to run dangerous command.

2.1. Mobile Agent

The mobile agent concept is illustrated in Figure 1. A client computer consists of an application environment, for example, OS/2 or Microsoft Windows, which contains one or more applications for interaction with a remote server. These applications are bound to an execution environment for mobile agents. The agent execution environment will also need to bind to various operating system functions, such as the memory manager, the timer, the file system and so forth. In particular the agent execution environment needs to bind to the message transport service in order to send and receive mobile agents via the communication infrastructure.

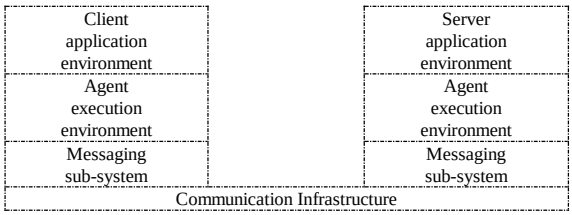

Figure 1: Conceptual model of mobile agent

Mobile agents encapsulate and transport data and code in a single mobile entity and not bound to the system where it begins execution. They are autonomous computer programs that can act in the interest of an entity, migrating between different network locations, executing tasks locally and continuing their execution at the point where they stopped before migration. Mobile agents can also have features like intelligence and ability to cooperate

Thus, an agent can be injected into the network to perform a task on a collection of distributed hosts, or perform tasks on the pervasive environment on behalf of an application, and then return the results of the requested actions on each host to the originating node. Figure 2 shows use of mobile agents which can reduce network connections.

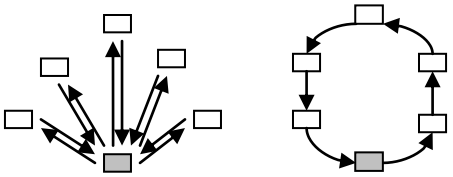

Figure 2: Difference between (a) non-mobile and (b) mobile connections

The use of mobile agents can bring several advantages to applications and their users: (1) reducing network traffic, as interactions can be carried out locally, independently of network latency; (2) asynchronous and decentralized execution, allowing the user to disconnect from the network when agents are performing a task; (3) ability to detect changes in the execution environment and react autonomously, simplifying the development of distributing systems that are more robust and fault tolerant.

2.2. Mobile Agent Pattern

This Controlling Access to Music Selection System on mobile agent architecture use *itinerary* design pattern of mobile agent.

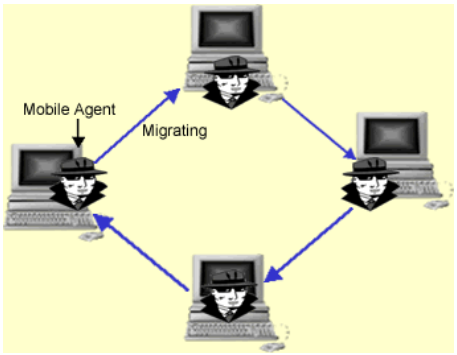

Figure 3: Itinerary pattern of mobile agent

Itinerary pattern provides a way to execute the migration of an agent, which will be responsible for executing a given task in remote hosts. The agent receives an itinerary on the source agency, indicating the sequence of agencies it should visit. Once in an agency, the agent executes its task locally and then continues on its itinerary. After visiting the last agency, the agent goes back to its source agency. Figure 3 illustrates this pattern.

3. Controlling Access to Music Selection System on Mobile Agent Architecture

This section describes use of mobile agent for controlling access of music selection. This system consists of several interacting agents, Vote Collector (VC), Vote Manager (VM) and Vote Administrator (VA) and Voters (VOs). The VA is responsible for registering candidate music list for elections and commissioning VM. The VC is a mobile agent mandated by a stationary VM agent to collect votes from stationary voting agents (VOs).

Use case diagram in Figure 4 shows the vote collection and reporting process, in which the VC collects votes from voters (VOs), and then forwards the votes to the VM which in turn forwards it to the

VA. The VA verifies the votes and delivers the results to the VM, and voters.

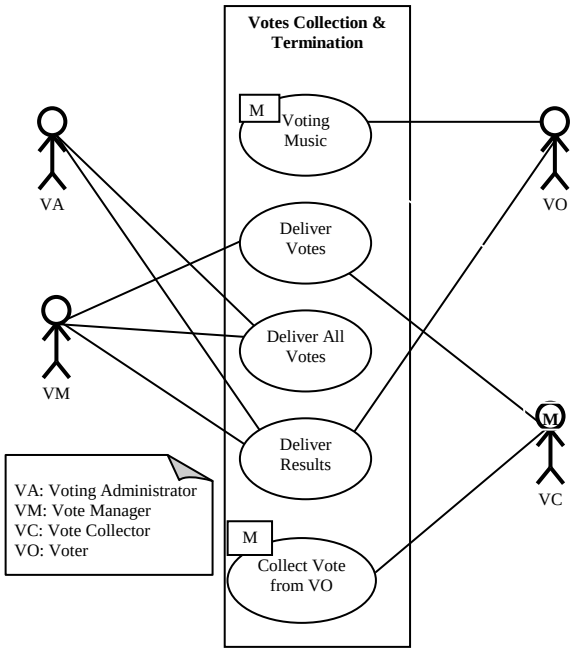

Figure 4: Use case diagram for music vote collection and termination

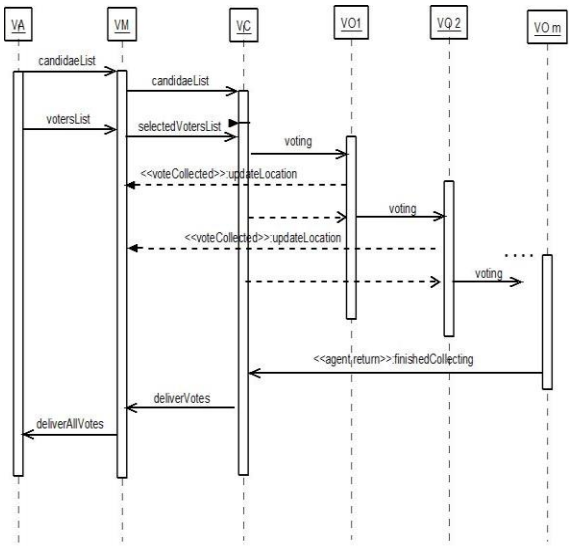

Figure 5: The sequence diagram of placing candidate music list

The sequence diagram in Figure 5 describes placing candidate music list. The list of candidate music and voters are sent by VA to the VM, which in turn forward the list to VC. Once the list is received by VO, the VO instantiate its stationary agent with user requested music. On election time, VC will visit the VOs on its list and get their votes, and once done, they return back to and reporting the collected votes to VM.

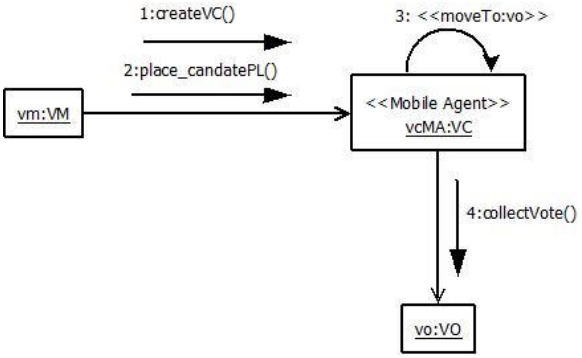

Figure 6: The collaboration diagram of placing candidate music list

The precedent Figure (Figure 6) describes the sequence diagram of the placing candidate playlist. Then to place candidate playlist:

- A VM agent creates VC mobile agent vcMA (1:createVC());
- Then, the vcMA invokes it to place the candidate playlist (2:place_candidatePL());
- The vcMA should moves to the platform of VO in order to finish its tasks (3:moveTo:VO);
- Finally, vcMA interacts locally with the VO in order to finish its tasks (4:collect_vote()).

4. System Design and Implementation

The mobile agent VC collect votes from VO as follow:

(i). VC gets candidate music list and voters list from VM.
(ii). VC goes to and collects votes from the VO_1.
(iii). Then VC goes to VO_2 and finally to VO_n.
(iv). When VC finish collections of votes, return back to and deliver votes to VM.

VM will dispatch the result of the votes to the VA. Finally, VA generates playlist according to the user selection/votes. This system is implemented as the following steps:

Step 1. VA agent generates and sends candidate playlist to VM agent.
Step 2. VM agent dispatches these lists to mobile agent VC.
Step 3. VC goes to each host and collects votes from one VO to other VOs agent which works on behalf of each user and bring the music suggestion from user.
Step 4. VC agent gathers the information from each VO agent.
Step 5. VC agent sends all collected information including selected music list to VM.
Step 6. VM forward candidate music list and voters to VA agent.
Step 7. VA agent generates playlist with most demand music for broadcast.

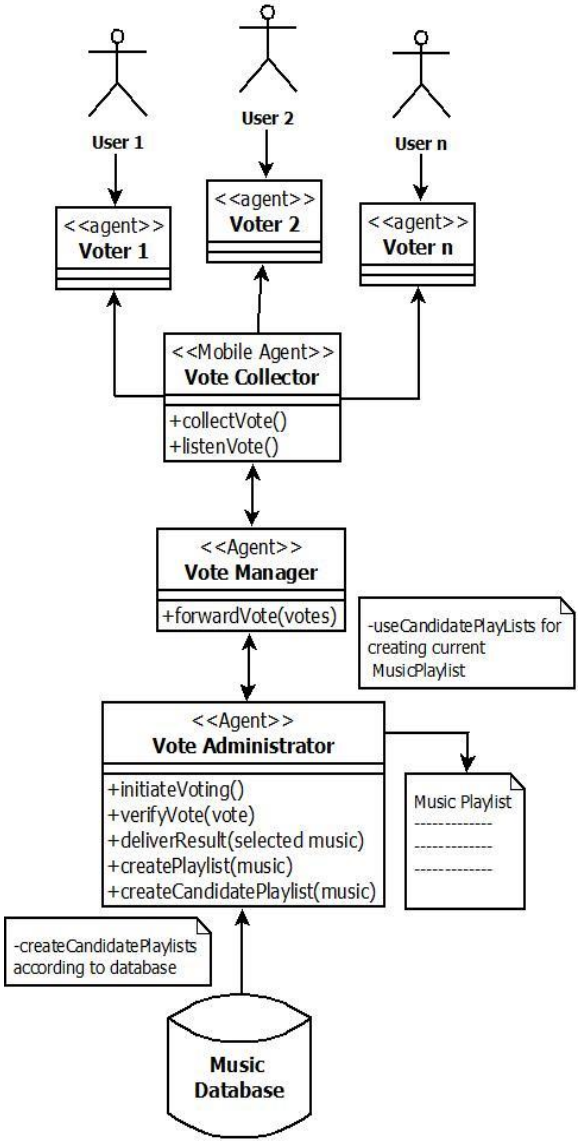

Figure 7: Controlling access to music selection system on mobile agent architecture

Admin create candidate music list for music selection process and ending the music selection process. Admin choose music tracks for candidate music list. When music selection start, mobile agent is created and collect vote from each user. Figure 8 shows the available location and visited location of mobile agent. For agent migration, the beforeMove () method is called at the starting location when the move operation has successfully completed, so that the moved agent instance on the destination container is about to be activated and the original agent instance is about to be stopped. However, as an immediate consequence, any information that must be transported by the agent to the new location has to be set before the doMove () method is called. For instance setting an agent attribute in the beforeMove () method will have no impact on the moved instance. The afterMove () method is called at the destination location as soon as the agent has arrived and its identity is in place.

```
void beforeMove()
{
     gui.dispose();
     gui.setVisible(false);
}

void afterMove()
{

// creates and shows the GUI
     gui = new MobileAgentGui(this);

//if the migration is via RMA the
     variable nextSite can be null.

if(nextSite != null){
 visitedLocations.addElementnextSite);

 for(inti=0;i<visitedLocations.size();i++)
   gui.addVisitedSite(
     (Location)visitedLocations.elementAt(
     i));
}
gui.setVisible(true);
}
```

The beforeMove () method is executed just before moving the agent to another location. The afterMove () method is executed as soon as the agent arrives to the new destination. It creates a new GUI and sets the list of visited locations and the list of available locations (via the behavior) in the GUI.

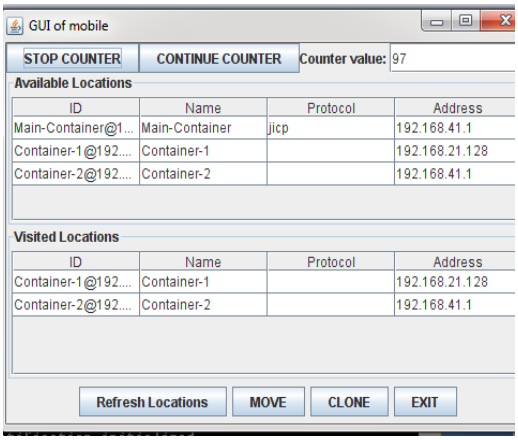

Figure 8: Mobile agent with available location and visited location

Admin check voting results and create music playlist for listening. When music selection starts, user can choose desired track and playback the playlist after voting end by admin. Figure 9 shows the flash player with voted songs as playlist.

Figure 9: Flash player with number of votes for each song

6. Limitations and Advantages

Mobile agents reduce the network load. The motto is simple: move the computations to the data rather than the data to the computations. They overcome network latency. Mobile agents offer a solution, because they can be dispatched from a central controller to act locally and directly execute the controller's directions. They encapsulate protocols. When data are exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data, respectively.

They execute asynchronously and autonomously. Often, mobile devices must rely on expensive or fragile network connections. Tasks that require a continuously open connection between a mobile device and a fixed network probably will not be economically or technically feasible. To solve this problem, tasks can be embedded into mobile agents, which can then be dispatched into the network. After being dispatched, the mobile agents become independent of the creating process and can operate asynchronously, and autonomously**.**

7. Evaluation Result

The experiment is repeated for every activity, with the locations on the same host and on different hosts. When the music selection is start mobile agent is created on server machine. On remote machine, remote container is created to migrate mobile agent from server machine and then mobile agent migrates to that host. Before moving to remote host, beforeMove() method is executed and communicate with other JADE (Java Agent Development Environment) platform from remote host. The afterMove () method is executed after arrived to remote host. This system was tested with the different locations on different hosts. The testing environments for a number of locations located on different hosts were four PCs : server hosted the Controller Agent and the JADE main container and the other with container for every host. Figure 10 shows communication of machines to migrate mobile agent. All PC's were connected to a 100Mbps network. The performance was measured for every activity based on the number of locations and the average time taken to complete the test.

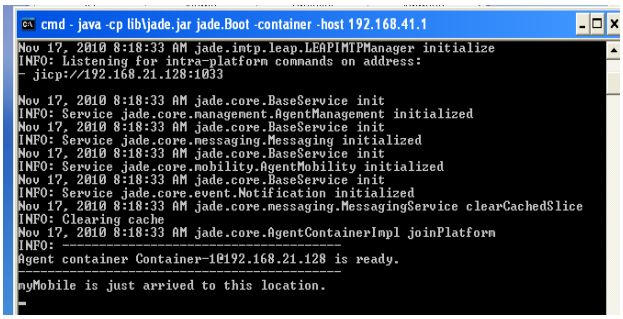

Figure 10: Mobile agent arrive to remote host

8. Conclusion

The main purpose of this system is to apply mobile agent architecture to the Music Selection System. The design of the system intends to automate the entire operations of music selection and helps administrative processes. The proposed mobile agent approach reduces human involvements in processes. This music selection system has many advantages over conventional systems.

- Encourages social activities and user participation
- All users can suggest their individual songs to the group
- Automated less user effort
- Flexible-high level of automation
- Ease of use
- Platform independent
- Location independent
- No need stand-alone media player, just need plug-in on web browser
- Support for disconnected mode of operation
- Reduce network connections

9. References

[1] A. Komninos, Neil Cameron, Colin Feron, and Richard Allan, Alistair Lindsay Glasgow Caledonian University 70 Cowcaddens Road Glasgow G4 0BA, UK + 44 141 331 3095andreas.komninos@gcal.ac.uk; AmbientDJ: *Enabling Interaction between People and Socially Aware Environments.*

[2] D. Sprague, University of Victoria 3800 Finnerty Road Victoria, BC, Canada dsprague@cs.uvic.ca Fuqu Wu University of Victoria 3800 Finnerty Road Victoria, BC, Canada fuquwu@cs.uvic.ca Melanie

Tory University of Victoria 3800 Finnerty Road Victoria, BC, Canada mtory@cs.uvic.ca**.** *Music Selection using the Party Vote Democratic Jukebox.*

[3] F. Sterling, Bob Tarr, Danko Nebesh, Mobile Agents, Department of Defense USA 2000. *Introduction to Mobile Agent Systems and Applications*.

[4] K. Saleh, C.El-Mirr, A. Mourtada and Y. Morad American university of Sharjah P.P.Box 26666, Sharjah, United Arab Emirates. Specifications of a *Mobile Electronic Voting System and a Mobile Agent Platform.*

[5] K. Miao, Lan Wang and Kenji Taguchi School of Computing, Leeds Metropolitan University Department of Computing, School of Informatics, University of Bradford. *Modeling Mobile Agent Applications in UML2 Activity Diagrams.*

[6] K. O'Hara,1, 2, Matthew Lipson1, 2, Marcel Jansen1, 2, Axel Unger1, Huw effries1, 2, Peter Macer11The Appliance Studio University Gate Park Row Bristol, UK 2University of Bristol Computer Science Department Mobile Bristol C/O FutureLab 1 Canons Road Bristol, UK. Jukola: *Democratic Music Choice in a Public Space*.

[7] L. Benedicenti and Songsiri Srisawangrat Faculty of Engineering University of Regina, Regina, SK, S4S 0A2, Canada. *Using Agents for a Participatory Collaborative Media Sharing Experience*.

[8] L. Emerson Ferreira de Araujo,Patricia Duarte de Lima Machado, Jorge Cesar Abbrantes de Figueiredo, Flavio Ronison Samoaio. *Implementing Mobile Agent Design Patterns in the JADE framework.*

[9] M. David, Nichols Hamilton, Department of Computer Science University of Waikato New Zealand {sallyjo, dmn}@cs.waikato.ac.nz.Exploring. *Social music behavior an investigation of music selection at parties.*

[10] S. Leitich, Markus Toth, University of Vienna, Institute for Distributed and multimedia Systems Liebiggasse 4/3-4, 1010 Vienna, Austria, stefan.leitich@univie.ac.at,a0025690@univie.ac.at. PublicDJ - *Music selection in public spaces as multiplayer game.*