Experimental Evaluation of Dynamic Exchange of Mobile Agent Components

Goh Matsubara

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 2000

Keywords: mobile agents, dynamic maintenance, remote maintenance, exchange of components.

1 Background

In this research, we propose and implement a mechanism that allows mobile agents to maintain by themselves.

Mobile agents are autonomous programs that can move from computer to computer with their internal condition. They can provide a convenient, efficient, and robust framework for implementing distributed applications. Several mobile agent systems have been released over the last few years.

Since mobile agents move to other computers through the Internet and they are often disconnected from their owner (i.e. user), mobile agents often run where their owner cannot control them directly. This makes it difficult to maintain active mobile agents. Consider the following situation.

1. For example, consider the search engine agent, which moves around web servers, and assume the agent analyze HTML files to make a database. When new type of web server is developed, the agent may not be able to analyze the new server. In the case of the normal application, its owner can maintain it by terminating its job and recompiling it. But, mobile agents may be in the place where an owner cannot access it directly. Then,in the worst case, the agent may not be able to continue its work.

2. Moreover, mobile agents depend on network strongly, and there are many applications using network connection. For example, in the case of proxy agents, many

connections are required from many clients, and then it is desirable that their communications are not disconnected while all clients are active. Now assume we would like to add to proxy agents a new facility to deal with some new protocol. Using traditional techniques, we need to terminate agents (i.e. disconnect their communications) to update proxy agents.

So, we must consider the following requirements.
Mobile agents should be maintained :

- without interactions with their owner.
- without terminating to keep connections.

Our idea to satisfy the above requirements is that we allow mobile agents to maintain themselves dynamically.

Fortunately, most mobile agent systems in recent years are implemented on the Java system, therefore dynamic exchange is easy to do. One reason why Java is used is that Java has flexible class load methods. Many mobile agent systems are realized by using this function. Our method also uses this function to implement dynamic exchange. It is, however difficult to treat with instances of old classes. After an old class is exchanged for the new class, the consistency of instance may collapse after the movement. We solve this problem by assuming that components have total order version and that when agent migrate to another host, the agent destroy the old instance.

2 Purpose

The purpose of this paper is to provide a mechanism that mobile agents maintains by exchanging parts of themselves dynamically. Especially, we pay attention to the distribution method of their components, which is suitable for the above mentioned requirements of mobile agents.

To show the efficiency and usefulness of our distribution method, we compare it with traditional methods: *push* method where the owner sends new components to agents, and *pull* method where agents bring new components from vaults. These traditional methods have two problems. One problem is that the number of network connections increases. The other is that there are no guarantee such vaults are always reachable via network in the case of mobile agents. In such a situation, mobile agents cannot obtain new components.

To show the feasibility of our distribution method, we implement experimentally it for a sample application.

3 Our Comparative Exchange Method

In this research, we focused a distribution method to exchange the mobile agent components.

So, we propose the method that components are distributed by comparing the components that agents hold each other and by exchanging them.

1. Every time an agent moves to a runtime system, the agent asks the other agents on the same system to send their component lists to the agent. The agent also send its component list to the other agents.
2. If an agent find a new component on the lists, the agent requires it from its owner.
3. The agent received it and then the agent replaces the corresponding old component with the new one by itself.

Figure 1: Comparative Exchange Method

4 Implementation

In this research, we propose and implement dynamic components exchange using "comparative exchange method".

We also actually make some experiments using the method, and we compare with the traditional methods (*push*/*pull*). In the experiments, we calculate the whole amount of network transfer, the distribution speed of the components, and so on.

We use Satoh's AgentSpace as the platform of our implementation, which is a mobile agent system implemented on Java VM (JDK 1.1). So, our system also requires JDK 1.1.

Java has a flexible class load method. Mobile agent systems are implemented by using this flexibility. In the case of AgentSpace, when agent migrates to another server, the agent recovers by using AgentClassLoader which was extended from ClassLoader. We also use this mechanism to build our dynamic exchange system.

5 Concluding Remarks

In this research, we propose and implement a mechanism that allows mobile agents to maintain by themselves. The maintenance method which using "comparative exchange method" has the following characteristics in comparison with traditional methods (such as *push*/*pull*). We actually implements example, which analyze some web servers, using that method. And , it was found out that there were the following characteristics as a tested result.

1. The number of network connection can be reduced.
2. The fixed informations can be reduced. Fixed informations are such as vaults place information. Therefore, it has a tolerance to the network topology as well that a connection with the vault becomes unstable.

3. When there are much number of the agents, the distribution of components can be efficiently, although the number of the agents is small, traditional method is more efficiently.