Migration Algorithm for Mobile Agents

Claudiu-Ionuţ POPÎRLAN

Computer Science Department, University of Craiova
A.I. Cuza Street, No. 13, 200585 Craiova, Romania
popirlan@gmail.com

Abstract. The concept of mobile agent is defined in [1]. There are autonomous objects that migrate from node to node of Internet and provide to user which have executed themselves using database or computation resources of clients connected by the network. To migrate the mobile agent, it will be needed a virtual place so-called the mobile agent system to support mobility. Mobile agent systems provide a virtual place for migrating mobile agents under our basic ideal condition that there are no faults on the systems or nodes, or include relevant protocols.
Keywords: Artificial Intelligence, Mobile Agents, Mobile Agents System, Network, Tracy, Mobile Technologies.
Math.Subject Classification 2000: 68T30, 68T40, 68T05

1 Introduction

The general structure of the paper is the following:

- The elaboration of migration algorithm: We propose migration algorithm with reordering and backward recovery of the paths to guarantee the migration of mobile agents. The proposed algorithm not only affords to avoid any faults of nodes or clients of mobile agents on network but also affects to agents' life span.
- An application: This uses the suggested algorithm and offers a strategy with the simple techniques of path reordering and backward recovery to migrate mobile agents.
- Open problems and future work. While a mobile agent is launched to specific nodes/clients according to relevant routing schedules, it is possible to happen some problems about migration of mobile agent if the host happens an accident within where the agent visits and executes. The protocol is for detecting and processing what occurs any fault on migrating mobile agents.

However, it does not provide to guarantee migration reliability of mobile agent. There is a simple protocol using transaction message queue, which is a procedure that the sender puts messages in the queue and receiver gets messages. For example, assume that there is an agent in input queue of a host and the node's error occurs before the agent moves to queue of next node. Then the agent is blocked until that the node is recovered. This situation differs from problem in client/server.

2 Algorithms for mobile agents

I consider an agent moves from a node to the consecutive node along the path:

$$X1 \rightarrow X2 \rightarrow \cdots \rightarrow X(k-1) \rightarrow Xk,$$

where Xi is a node from the wireless network. As an agent may visit the same node several times Xi and Xj ($1 \leq i, j \leq k$) may denote the same or different nodes. Assume further that an agent is stored in a repository when it is accepted by the agent system for execution. Each node, except Xk, performs the following sequence of operations on Transaction Ti:

1. *Get(agent)* : removes an agent from the node's repository.
2. *Execute(agent)* : performs the received agent locally.
3. *Put(agent)* : places it on the repository of the host that will be visited the right next time.

These three operations are performed within a transaction and hence consisted of the atomic unit of work.

I will present,in this section, the following new algorithm result:

* Migration Algorithm

  I will suppose that an agent migrates and executes from node X1 to X2 sequentially, but it is blocked at the client of node X2 until the node X3 is recovered. If the node X3 was not recovered, the agent may be orphaned or destroyed by the particular client. To solve this situation has the agent to skip the faulty node X3 that includes on the migration path, and to move the address of node X3 back to the last one of the migration path. And then, the node X2 successfully connects the next other node X4 without any fault. Node X4 also has a particular fault.

  Therefore, node X2 hops the right consecutive next one of node X4. As the same method is also applied to other nodes, the agent's migration path has reordered. That is, despite of any particular faulty nodes, the agent tries to connect subsequent nodes for the migration touring. This solution changes the previous arranged migration path by connecting normal nodes except that some nodes have the particular fault. Afterward, the agent retries to connect each certain fault node after it waits for the timestamp assigned by the mobile agent system. If the certain faulty node is recovered by the timestamp, the agent will succeed in migrating to the destination node. Otherwise, the address of the faulty node will be discarded. Since the agent may be able to loophole, we will give a restriction against the number of reconnection times.

  The structure of a migration algorithm is:

  Migration algorithm

  for (each agent's routing-table)

```
begin
  Extract a target address and fail checked information;
if (no more a target address)
  Backward multicasts signal to successful target nodes;
if (exist a fail checked address)
begin
  times or not
  // check whether connect more than two times or not
  Wait the agent during some system timestamp;
  Try to connect Socket to the address;
if (success)
begin
  Call go Agent;
  Exit;
end
else
begin
  Notify to user the address is unavailable;
  Ignore the address;
end
end
else
if (not a fail checked address)
begin
  Try to connect Socket to the destination node;
if (success)
begin
  Call go Agent;
  Exit;
end
else
begin
  Notify to user;
  Move the current failed address to last in the routing-table;
  Set the fail checked information;
  end
end
end
```

Figure 1 shows that a whole rearranged migration path for the mobile agent be changed by this strategy:

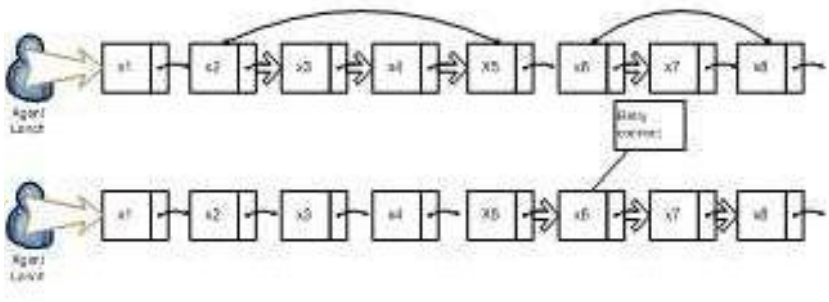

Fig. 1. The reordering before and after meet with faulty nodes.

3 An application which uses the migration algorithm

I implemented the proposed algorithms in TRACY [2], a model of mobile agent system based on Java language for system independent platform. The TRACY Toolkit uses the standard Java (Version 1.4) execution environment without any modifications, neither of the virtual machine nor of any of the core Java APIs. Large parts of TRACY are free software for non-commercial purposes distributed by the University of Jena.

The results obtained can improve effectively the problem of performance and networks overhead due to the imposed characteristics of distributed architecture since a mobile agent offers not only the migration reliability and transparency for mobile agent as autonomously as possible but also computing environment which is capable of distributed processing with mobile objects.

Mobile agents (in our case, specialised agents) autonomously collect usage data from all users and "learn" user habits using LRS(Longest Repeating Subsequence) [8] model. Helper agents then predict the next action to be performed by the user, and display available actions in order of probability in the specialised toolbar (see *Figure 3*). Additionally, agents cooperate and adjust user interface to suite users preference.

The proposed algorithm can be used in wireless network to obtain an efficiency of communications (the nodes reordering.

4 Conclusions and future work

HE following open problems arise from this paper:

- Comparative study concerning these two algorithms and other migration algorithms [3];
- Study the case when the agent has toured for all nodes having no faults before that it does re-connect with the faulty nodes.

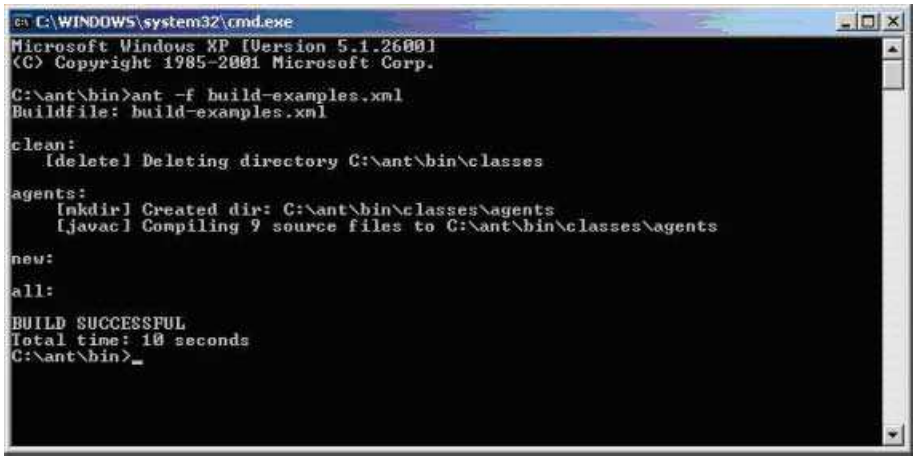

Fig. 2. A screen shot of building the algorithm in TRACY.

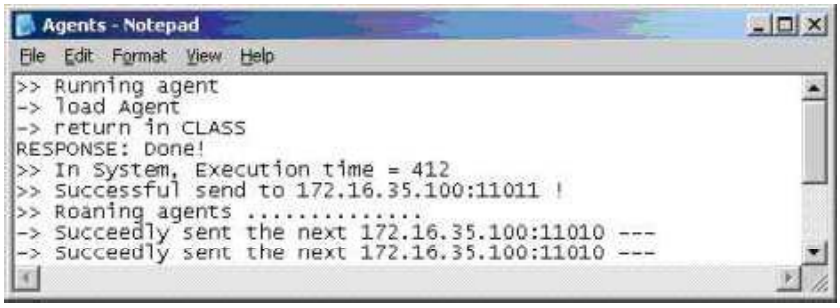

Fig. 3. A screen shot of executing the algorithm with TRACY.

In this paper we introduce the Migration Algorithm to ensure the migration of mobile agents in networks. The proposed scheme not only affords to avoid any faults of nodes or clients of mobile agents on network but also affects to agents' life span. All presented techniques have been implemented in TRACY. Therefore, TRACY can improve effectively the problem of performance and networks overhead due to the imposed characteristics of distributed architecture since a mobile agent offers not only the migration reliability and transparency for mobile agent as autonomously as possible but also computing environment which is capable of distributed processing with mobile objects.

References

[1] **P. Braun , W. Rossak**:- Mobile Agents: Concepts, Mobility Models, & the Tracy Toolkit, Elsevier Inc. (USA) and dpunkt.verlag (Germany), 2005

[2] ***:- The Tracy web page http://wiki.tracy.informatik.uni-jena.de/mobileagents/tiki-index.php

[3] **J. Baumann**:- Mobile Agents: Control Algorithms, Lecture Notes in Computer Science, Springer

[4] **Cl. Popîrlan, Cr. Popîrlan**:- Mobile Agents communication for knowledge representation, The 11th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2007, July 8-11, 2007 Orlando, Florida, USA, 231-238

[5] **Cl. Popîrlan, Cr. Popîrlan**:- Using Mobile Agents in User Interfaces Functionality, Research Notes in Artificial Intelligence and Digital Communications, 6-nd Romanian Conference on Artificial Intelligence and Digital Communications, Thessaloniki, Greece, August 2006, Vol.106, 62-68

[6] **M. Ghindeanu, Cr. Popîrlan**:- A Natural Language Processing System using Java-Prolog Technology, Vol.105, 5-nd Romanian Conference on Artificial Intelligence and Digital Communications, Craiova, June 2005

[7] **Cl. Popîrlan**:- A Java Implementation of Modeling Results About Stratified Graphs, Research Notes in Artificial Intelligence and Digital Communications, Vol.104, 4-nd Romanian Conference on Artificial Intelligence and Digital Communications, Craiova, June 2004, 31-38

[8] ***:- Java Remote Method Invocation, http://java.sun.com/products/jdk/rmi/

[9] ***:- XIML (eXtensible Interface Markup Language), http://www.ximl.org/

[10] ***:- Distributed Objects & Components: Mobile Agents, http://www.cetuslinks. org/oo_mobile_agents.html

[11] ***:- Foundation for Intelligent Physical Agents, http://www.fipa.org