Optimized Mobile Agent Migration Overcoming A Unique Deadlock Scenario In A Heterogeneous Environment

Aditya Vikram Bhunja
Final Year Student,
Bachelor of Computer Applications
Institute of Engineering & Management, Kolkata

Abstract

This paper proposes a new framework for mobile agents which would decrease network load in cyber net while dealing with a deadlock situation due to the heterogeneous environment. Some of the security issues and their solutions are also discussed to perform a secure and successful migration.

1. Introduction

Before discussing about Mobile Agents we should know about the term Agent, it means an individual representing another for a certain task. So in computer science, Agents (synonymously called Software Agents) are computer programs which resides and stays attached in the system itself to perform tasks on behalf of the user. So as in definition of Mobile Agents we can say that they are Software Agents with the gift of mobility. This means that it can migrate from system to system with its execution code and status. It came into existence as an improvement of the traditional network system which is done via RPC (Remote Procedural Call). The main advantage of Mobile Agent technique over the traditional technique is the reduction of network load.

In a distributed system, for completing a task there are multiples number of times information will moves form one node to another node. This is especially true when security measures are enabled. So that the result is a lot of network traffic and size of data will increase because, security thread will add every time when data is sent. Mobile agents allow us to dispatch all information of the task at a time when agent arrives at destination host the interactions can take place locally rather than transferred over the network. The objective of mobile agent bases communication is to move the computations to the data rather than the data to the computations.

2. Life-Cycle of a Mobile Agent

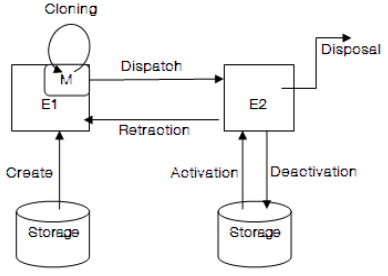

Figure 1 Life Cycle

As shown in Figure 1, the life cycle steps are as follows:

1. A mobile agent M is *created* in the Home Environment E1.
2. After execution the agent is *cloned* and is dispatched to Host Environment E
3. The *cloned* copy executes on E2.
4. After execution, E2 sends the mobile agent M received back to E1. The agent of E2 is then *disposed*.
5. E1 *retracts* the agent M and the data brought by the agent is analyzed.

From this we observe that a mobile agent experiences the following events in its life cycle:

Creation: a brand new agent is born and its state is initialized.

Dispatch: an agent travels to a new host.

Cloning: a twin agent is born and the current state of the original is duplicated in the clone.

Deactivation: an agent is put to sleep and its state is stored on a disk of the host.

Activation: a deactivated agent is brought back to life and its state is restored from disk.

Retraction: an agent is brought back from a remote host along with its state to the home machine.

Disposal: an agent is terminated and its state is lost forever.

3. The Problem

3.1. Problem Statement

Migration of a Mobile agent in a distributed network system can cause a lot of unexpected problems. The

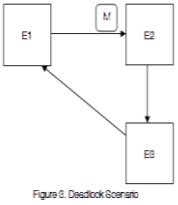

Figure 2. Deadlock Scenario

problems dealt in this paper can be categorized into two parts:-

1) A deadlock scenario due to heterogeneous environment.
2) The issue of secure and successful migration across the network in the proposed framework.

3.2. Problem Description

The problem consists of a minimum of three heterogeneous machines or environments through which the mobile agent migrates. Namely we take the three environments as E1, E2 and E3. Here E1 is the home machine and E2, E3 are the host machines. Now in normal mode of migration, the path is shown in Figure 2.

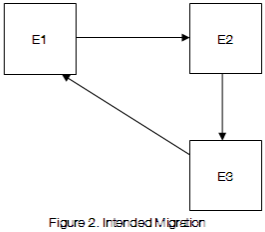

Figure 2. Intended Migration

Now, as being the home machine E1 creates a mobile agent and dispatches it to E2 as shown in Figure 3 where M symbolizes the Mobile Agent. But due to the heterogeneous natures of E1 and E2, they agent is not able to enter E2. This is due to the first rule of interoperability which states that an agent incompatible of the host environment should not be allowed to enter the environment. The home machine cannot even *retract* the agent since it is not into a particular environment and is suspended in mid cyber-space. As a result, the agent now cannot move backward to E1 or forward to E3 which requires MA to complete its task. Now this creates a deadlock scenario for the network.

4. The Solution

4.1. Solution Assumptions

- All RPC calls made by one node to other nodes are secure.
- All nodes in the network framework are trusted and non-malicious.
- The migrating mobile agent is non-malicious.

4.2. Solution Components

- E1,E2,E3-Heterogeneous environments.
- Code Server- Where the codes of the mobile agent of home machine are stored.
- Agent Blueprint- It is the skeletal structure of the programming code and functionalities of the mobile agent to be sent. In other words, it is an implement-tation independent descript-tion of the agent.
- Class Loader- It is a piece of software which creates an empty mobile agent using the blueprint and is also

responsible of loading codes (downloaded from the code server) into the empty agent.

- Authenticator- It is a token that is sent along with the mobile agent so they at least can enter into the architecture of the host machine.
- Forward Pointer Scheming- It is a procedure of the home machine to send the host machines the node address of the agent's next hop according to the routing table.
- Backward Pointer Scheming- It is a procedure of the home machine to send the host machines the node address of the agent's previous hop according to the routing table.

4.3. Solution Description

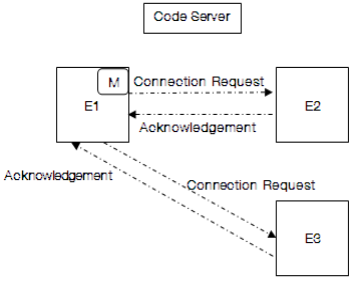

Figure 4. Connection Request

Firstly a mobile agent is created at the home machine. Then the home machine makes connection request to the host machines. After receiving acknowledgment from the host machines (as shown in Figure 4) in a stipulated time a routing table is made in the home machine and passed to the code server.

Now the made *agent's information* is sent to the host machines. The *agent information* contains the following details about the mobile agent:

- Agent's blueprint
- Home machine's authenticator.
- Address of next hop (forward pointer scheming).
- Address of previous hop (backward pointer scheming).

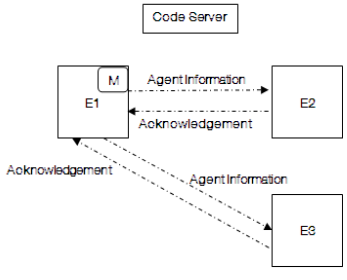

Figure 5. Passing Agent Information

After receiving the information, the hosts again send back an acknowledgement. This procedure is diagrammatically described in Figure 5.

Now the agent is now dispatched from E1 to E2. In this case, the agent carries the following information with itself:

a) Home machine's address
b) Home machine's authentic-cator.
c) Destination host machine's address(E2's address).
d) Previous hop address (also Home Machine's Address).
e) Data.

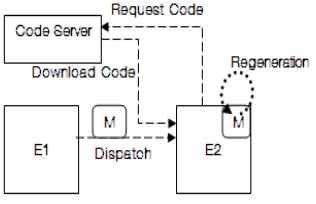

Figure 6. Migration from E1 to E2

After reaching E2, the agent's authenticator and its previous node address is compared with the received authenticator and previous hop address from E1. If even one of them do not match then its means that the agent has been compromised and the agent is sent back to the home machine. If both information match then a request for the mobile agent code is done to the code server of the home machine E1. If the request is granted then a new mobile agent is re-created or regenerated within the host machine according to its configuration via the Class Loader. Now the new agent copies the data from the received agent and executes in the host environment. This whole process is well described in Figure 6.

After execution, the new mobile agent is cloned and dispatched from the current environment while appending the next hop address in it. After it has been dispatched, a signal is sent to the home machine of its dispatch. The home machine then sends an auto-destruct command to both the sent mobile agent and the new mobile agent which still resides in E2. The new mobile contains the following components:

a) New updated data.
b) Home machine's address
c) Home machine's authentic-cator.
d) Destination host machine's address (E3's address).
e) Previous hop address (E2's Address).

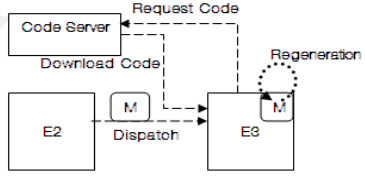

Figure 7. Migration from E2 to E3

On reaching E3, as shown in Figure 7, the same methodologies are used to tackle the execution and dispatch.

On reaching E1, the environment again regenerates the agent (as shown in Figure 8) and destroys the received agent after extracting its data.

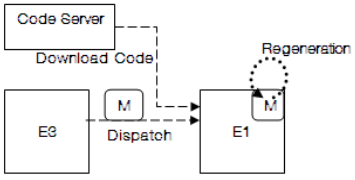

Figure 8. Migration from E3 to E1

5. Modification in the Life Cycle of Mobile Agents

The above solution framework does changes the life cycle in a small but significant way. Figure 9 shows the change vividly below.

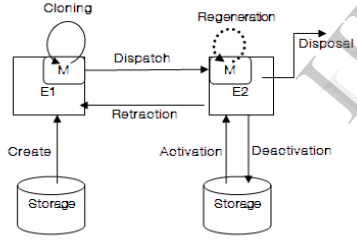

Figure 9. Modified Life Cycle

6. Drawbacks

- Regeneration is done even if both environments are homogenous in nature.
- All security issues like eavesdropping, masquerading, etc. are not covered.

7. Conclusion

Agents, and in particular mobile agents, offer a means for application developers to build distributed applications. Mobility of agents is often required for various reasons, notably performance. However, mobility of agents is usually limited to hosts running the same agent platform and that have the same (virtual) machine architecture. In other words, it is often restrained to a homogeneous environment.

The proposed framework supports heterogeneous mobility, offering an agent maximum flexibility with respect to where it wants to go. To summarize the whole paper, we can sa that a blueprint of an agent's functionality is transported, together with information on the agent's state. At its destination, an agent factory regenerates the executable code of the agent on the basis of its blueprint. An agent may then restore its state and resume execution. In other words, blueprints offer a language and agent platform independent virtual machine that allows for heterogeneous migration.

8. References

[1] D. B. Lange and M. Oshima, Seven good reasons for mobile agents, Communications of the ACM, 42 (1999), pp. 88–89.

[2] L. Magnin, T. V. Pham, A. Dury, N. Besson, and A. Thiefaine, Our guest agents are welcome to your agent platforms, in Proceedings of the ACM Symposium on Applied Computing (SAC 2002), Madrid, Spain, Mar. 2002, pp. 107–114.

[3] D. C. Smith, A. Cypher and J. Spohrer (1994) "Programming Agents without a programming language" Communications of the ACM 37 (7) pp 55-67.

[4] J. White. "Mobile Agents".In J. M. Bradshaw (editor), Software Agents. AAAI Press/MIT Press, 1997, pp. 437-472.

[5] P. Morreale. "Agents on the move". In IEEE Spectrum, April 1998, pp. 34-41.

[6] B.J. Overreinder, D.R.A. De Groot, N.J.E. Wijngaards and F.M.T. Brazier (2006), "Generative Mobile Agent Migration in Heterogeneous Environments" in Scalable Computing: Practice and Experience, volume 7, number 4, pages 89-99.

[7] Xinyu Feng (2002), "Design and Analysis of Mobile Agent Communication Protocols", Institute of Computer Software Nanjing University, China