Hierarchal Object Oriented Fault Tolerant Secured and Atomic Mobile Agent Model.

Mayank Aggarwal[1] and Nipur[2]

[1] Department of Computer Science and Engineering, Gurukul Kangri University, Haridwar, Uttarakhand 249404, India

[2] Department of Computer Science, Kanya Gurukul Mahavidyalaya, Dehradun, Uttarakhand ZIP/Zone, India

Abstract

Mobile Agents are soft wares migrating from one node to another to fulfill the task of its owner. Mobility introduces two major challenges in front of mobile agent namely reliability and security. As the agent moves from one node to another the goal to complete its task safely is difficult to achieve. Mobile agents are no longer a theoretical concept, much architecture for their realizations have been proposed. However, it has to be confirmed that any failures (machine or agent) do not lead to blocking of agent together with the security issues. This paper proposes a model which deals with both the problems; Fault Tolerance and Security, further it also adds atomicity to mobile agents execution i.e. either all the goals are achieved or none is achieved. This paper proposes a Hierarchal model which uses the concepts of object oriented technology, grouping, atomicity and authentication to deal with the blocking problem and security issues.

Keywords: Mobile Agent, Blocking, Atomicity, Object Oriented, Grouping.

1. Introduction

Mobile agents are software that acts autonomously on behalf of a user and migrate through a network of heterogeneous machines [1].The advantage for using mobile agent technology is that interaction cost for the agent-owner is remarkably reduced since after leaving its owner the agent migrates from one host to the next autonomously [9]. Still, even today, only few real applications rely on mobile agent technology might be due to the lack of transaction support for mobile agents [6]. When a mobile agent migrates from one host to another variety of faults may occur, it may be a system crash, corruption of agent, failure of platform, link failure etc but the objective should be that execution is not blocked[7]. To evade from blocking problem replication was introduced which provided another challenge of exactly once problem which was tackled in [15]. The paper provides a solution for blocking problem by grouping mobile agent platform which provide same type of services. Security remains the major hurdle in the field of mobile agent, as the agent has to be executed on hosts other than its owner chances of the host being malicious is very prominent. A lot of research issues in the security of mobile agent are discussed in [4, 5]. The model proposed a hierarchal structure with authorization process involved at every step to make the system secure together with a trusted hardware approach for final execution. Atomicity of an agent means that if the owner wants more than one task to be done, and all the tasks are interrelated than the transactions should be committed if and only if all the tasks have been successfully carried out. For example an agent whose task is to buy an airline ticket, book a hotel room, and rent a car at the flight destination. The agent owner, i.e., the person or application that has created the agent, naturally wants all three operations to succeed or none at all. Clearly, the rental car at the destination is of no use if no flight to the destination is available. On the other hand, the airline ticket may be useless if no rental car is available. The mobile agent's operations thus need to execute atomically. The proposed model incorporates atomicity by final commitment to be done separately by trust server when it receives results from all the groups

2. Background

Security of mobile agents involves two main issues, protecting agent against platform and protecting platform from agent. A lot of research has been done to make the mobile agent system secure. Techniques like Software-based Fault Isolation, Safe Code Interpretation, Signed Code, State Appraisal, Path Histories, and Proof Carrying Code has been used to protect the platform [18]. Similarly, various security mechanisms for protecting agent themselves are Partial Result Encapsulation, Mutual Itinerary Recording, Itinerary Recording with Replication and Voting, Execution Tracing, Environmental Key

Generation, Computing with Encrypted Functions, and Obfuscated Code [16, 17] .This proposed model ensures security by several measures like mutual authentication between agent and host, authentication at each level, trusted hardware and the concept of path history [2]. The IEEE83 defines fault tolerance as "The ability of a system or component to continue normal operation despite the presence of hardware or software faults." There are several fault tolerance approaches [7] like Spatial Replication, Primary Backup Protocols, Active Clients Primary Backup Model these approaches result in violation of exactly once property. Some other approaches which preserve exactly once property are Agent execution model, Enhanced Agent execution model, Voting Protocol, Rear Guards etc. The proposed model makes the system fault tolerant (non blocking) by making groups of similar mobile agent platform. The concept of Hierarchal model has been discussed in [10, 12, and 14]. Blocking occurs [7], if the failure of a single component prevents the agent from continuing its execution. In contrast, the non-blocking property ensures that the mobile agent execution can make progress any time. A non-blocking transactional mobile agent execution has the important advantage, that it can make progress despite failures. In a blocking agent execution, progress is only possible when the failed component has recovered. The proposed model incorporates non blocking property by grouping of mobile agent platforms and ensures atomicity by doing the final commitment at the trust server. As far as we have surveyed this is the first model which incorporates security, fault tolerance and also atomicity for transactional mobile agent.

3. Model

Security of mobile agents involve

The proposed model has four main components:

- Trust server
- Local Network Server
- Mobile Agent System (Group)
- Mobile Agent (Object)

2.1 Trust server

Trust server is the topmost layer of the model. It is responsible for mobile agent authentication and commitment. It receives the agent from the group incharge to be migrated to some other group. On receiving the agent it first decrypts the header, which contains the agent id, source id, destination id, path history (if any). The decryption is done by the private key of the server itself. It then checks for any threat in the decrypted header by comparing the information with its knowledge base if the agent is not safe it is put in the prison. After proper authentication, trust server prepares the agent to migrate further. First of all it saves the computed result (if any) in its knowledge base, and then it encrypts the agent with the public key of local network server and sends it to local server.

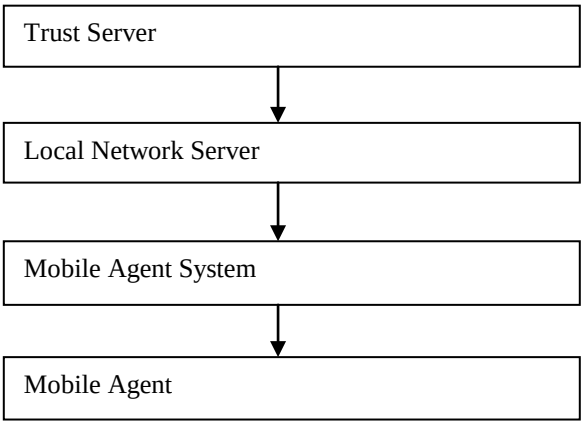

Fig. 1 Hierarchal Model

Further to it, it is also responsible to achieve atomicity, when it has received all the result; it commits all the computed transaction in a safe place protected by the trusted hardware.

2.2 Local Network Server

It receives the agent from the trust server and transfers it to the respective group incharge. On receiving the agent it decrypts the agent, compares the information with its security base and if all is found to be authentic then migrate the agent to the respective group incharge whose id is given in the header. Before sending the agent, it encrypts the agent with the public key of the respective group incharge.

2.3 Group Incharge

The group incharge on receiving the agent decrypts the agent with its private key. Check for the authenticity of the agent and also do the mutual authentication. The group is made of more than one mobile agent platform doing the same type of service. The incharge depending on the load on its members transfer the agent to one of its member for execution. On finishing the task agent returns back o group incharge which encrypt it with public key of trust server

and send it to trust server for further execution. It is designed as System Net.

2.4 Mobile Agent

Mobile agent itself is defined as on object of the Mobile agent platform, which is defined as Object Net of the System Net. The agent is defined as an object having an interface to communicate with outside world, knowledge base, header, path history.

4. Grouping

Blocking is one of the major problems in mobile agent. Many solutions have been presented before to avoid blocking of agents [15]. This paper has done grouping of mobile agent platforms to avoid blocking. A mobile agent submitted at one host within the group can be executed by any host of the group. Grouping can be done based on different criteria like

- Services offered
- Capability of hosts

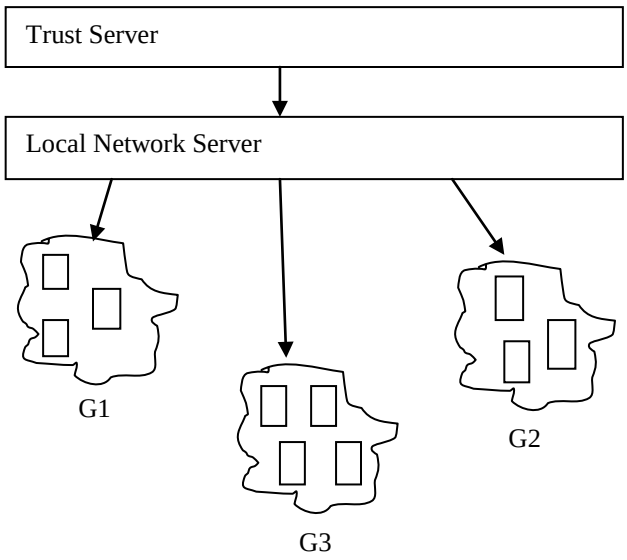

Fig. 2 Detailed Model Showing Groups

- Robustness of the hosts
- Authority to access shared data
- Communication route available between hosts.

This paper has grouped hosts on the basis of the services offered. One of the hosts is decided as incharge of the group. The incharge decide to whom the agent should be sent next within the group. If a particular host fails, agent continues its execution to some other host within the group as decided by the incharge. The problem of replication is avoided as agent is executed on only one host at a time. There may be a case in which incharge itself fails, in that case some other host is nominated as incharge by all other hosts in the group.

5. Hierarchical and Object Oriented Approach

In order to make the model secure and reliable, hierarchal and object oriented approach is used in designing of the model. There are four basic levels in the model, on the top trust sever, below it local server, then mobile agent platform in the form of groups and finally mobile agent which is defined as an object of the mobile agent platform. The hierarchical approach reduces network traffic as well as communication delay. Security is achieved as encryption-decryption is done at each level. Migration of agent from one level to another requires encryption of the agent by the public key of the receiver, which can be decrypted only by the receiver by its private key. Defining agent as an object adds to the security of the agent, it is modified only at the authenticated mobile agent platform. Object oriented approach also supports mobility of the agent.

6. Algorithm

The model has mainly four algorithms one for each level.

6.1 Trust Server Algorithm:

Step 1: Receive the agent

Step 2: Decrypt the agent

Step 3: Authenticate the agent if authenticated go to step 4 else put the agent in prison and exit.

Step 4: Collect the partial results in its knowledge base.

Step 5: If all the results have been collected go to step 6 else go to step 7.

Step 6: Commit the transactions in the assigned safe host.

Step 7: Encrypt the agent with public key of local server.
Step 7: Transfer the agent to local server.

6.2 Local Server Algorithm:

Step 1: Receive the agent

Step 2: Decrypt the agent with private key of local server.

Step 3: Authenticate the agent if authenticated go to step 4 else put the agent in prison and exit.

Step 4: Encrypt the agent with public key of group incharge.

Step 5: Transfer the agent to group incharge.

6.3 Group Incharge Algorithm:

Step 1: Receive the agent

Step 2: Decrypt the agent with its private key.

Step 3: Authenticate the agent if authenticated go to step 4 else put the agent in prison and exit.

Step 4: Search for member host with minimum load

Step 5: Send agent to host for execution

Step 6: If the selected host fails, search for another host and send agent to next selected host.

Step 7: Receive the agent back after it completes its execution.

Step 8: Update the path history, source id, and destination id.

Step 9: Encrypt the agent with public key of trust server.

Step 10: Send the agent to global network.

6.4 Owner Algoritm (Agent):

Step 1: Create the agent

Step 2: Crate header having the source id, destination id, agent id.

Step 3: Encrypt the agent with the public key of Trust Server.

Step 4: Transfer it to trust server.

7. Workflow of Model

The proposed model goes through a sequence of steps to achieve its goal. The model works as described in the above algorithms. The work flow is shown in Figure 3, with numbers showing the sequence of flow of agent.

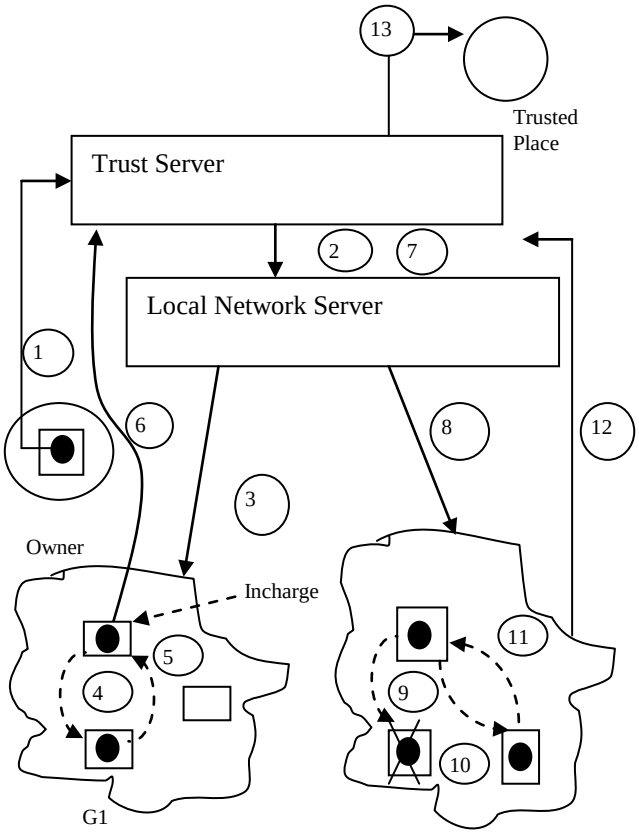

Fig. 3 Workflow of the model

8. Conclusion

The proposed model gives a unique way to implement a secured mobile agent model which tackles the problems of fault also. The simulation of the model is to be done, which is also under process by us using CPN tools [3].

References

[1] Antonio Corradi, Marco Cremonini, Rebecca Montanari, and Cesare Stefanelli, "Mobile agents integrity for electronic commerce applications," Information Systems, 24(6), 1999.

[2] Cao, Chun and Lu Jian, "Path-history-based access control for mobile agents," International Journal of Parallel, Emergent and Distributed Systems, vol 21: 3, pp 215 — 225 , 2006.

[3] CPN Tools website:www.daimi.au.dk/CPNtools

[4] Dirk Westhoff, Markus Schneider, Claus Unger, and Firoz Kaderali, "Methods for protecting a mobile agent's route," Information Security, Second International Workshop (ISW'99), number 1729 in LNCS. Springer Verlag, 1999.
[5] Dirk Westhoff, Markus Schneider, Claus Unger, and Firoz Kaderali, "Protecting a mobile agent's route against collusions," Selected Areas in Cryptography, 6th Annual International Workshop (SAC'99), number 1758 in LNCS. Springer Verlag, 2000.
[6] Dong Chun Lee and Jeom Goo Kim, "Adaptive migration strategy for mobile agents on internet," Technologies for E-Services (TES 2001), Second International Workshop, Proceedings, number 2193 in LNCS. Springer Verlag, 2001.
[7] Fred B. Schneider "Towards fault-tolerant and secure agentry," Distributed Algorithms, 11th International Workshop (WDAG'97), Proceedings, number 1320 in LNCS. Springer Verlag, 1997.
[8] Garrigues, C., et al, "Promoting the development of secure mobile agent applications," J. Syst. Software (2009), doi:10.1016/ j.jss.2009.11.001
[9] Giovanni Vigna, "Cryptographic traces for mobile agents," G. Vigna, editor, Mobile Agents and Security, number 1419 in LNCS. Springer Verlag, 1998.
[10] H.Pathank, K.Garg, "CPN Model for Hierarchical fault tolerance protool for mobile agent systems," 16th IEEE International conference on netwroks, 2008.
[11] Lotfi Benachenhou, Samuel Pierre, "Protection of a mobile agent with a reference clone," Elsevier , Computer Communications , vol 29, pp. 268-278, 2006.
[12] N.Desai, K.Garg, M.Mishra, "Modelling Hierrarchical Mobile Agent Security Prorotocol Using CP Nets," Springer-Verlag Berlin Heidelberg, LNCS 4873, pp 637-649, 2007.
[13] Price, Sean M., "Host-Based Security Challenges and Controls: A Survey of Contemporary Research," Information Security Journal: A Global Perspective, vol 17: 4, pp 170 — 178, 2008.
[14] Satoh I, "A hierarichal model of mobile agents and its multimedia applications," Parallel and distributed systems : Workshops, Seventh International conference , Japan, 2000.
[15] Stefan Pleisch, Andre Schiper, "Modeling Fault-Tolerant Mobile Agent Execution as a Sequence of Agreement Problems," Proceedings of the 19th IEEE Symposium on Reliable Distributed System (SRDS) p.p. 11-20, 2000.
[16] Tomas Sander and Christian F. Tschudin, "Protecting mobile agents against malicious hosts," in G. Vigna, editor, Mobile Agents and Security, number 1419 in LNCS. Springer Verlag, 1998.
[17] Venkatesan S, et al., "Advanced mobile agent security models for code integrity and malicious availability check.," J Network Comput Appl, doi:10.1016/j.jnca.2010.03.010 ,2010 .
[18] W.A.Jansen, "Countermeasures for mobile agent security," Elsevier, Computer Communications, vol. 23 , pp. 1667-1676 , 2000.