ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 2, February 2014

Malicious Host and Problem of Blocking for Mobile Agent: Proposed Solution

Dr. Heman Pathak

Associate Professor, Department of Computer Science, GKV, Haridwar, Uttarakhand, India[1]

Abstract: Mobile Agent (MA) is a software programs that lives in computer networks, performing its computations and moving from host to host as necessary to fulfil user goals. Autonomous behaviour of MA and the malicious environment of the internet give rise to various important security issues related with both MA and its host. Various researchers working in the areas have identified various threats and their effects. During its life cycle a MA moves from one host to other. If MA is malicious or if the executing environment i.e. host is malicious they both can influence each other and can harm in many ways. If the MA wishes to visit a host which has been detected as malicious then it should not migrate to that host. In this case MA will be blocked at its current location. This paper discusses the problem of blocking for MA for malicious host and proposes a group based layered architecture to mask the malicious host from rest of the network. Proposed architecture is centralized at one level and distributed at other. It divides the open network like internet into regions and hosts in each region are then grouped on the basis of services they offered. One of the host acts as group in-charge. This group appears as a single host to the other parts of the network. In case, a host is found malicious, load of that host is distributed among other trusted group members and recovery procedure for host started.

I. INTRODUCTION

Mobile Agent (MAs) is autonomous objects that are able to migrate from node to node in a computer network. The ability to roam the net is provided by a middleware platform, a Mobile Agent System (MAS). The use of MAs has been proposed for many application areas, including electronic commerce, systems management, or active messaging [1][2]. All these applications require a MA to be executed reliably. Before MA applications begin to appear on a large scale, MASs need to provide infrastructure services to facilitate MA development. Among these are security, fault tolerance, location management and transaction support.

If itinerary (The set of hosts to be visited by a MA during its life cycle) of MA is static (entirely defined at the MA source and does not change during its execution) and order of visiting hosts is fixed then MA will be blocked if the target host at any stage of its life cycle is detected malicious till the host recovers and become trust worthy again. Blocking MA executions are undesirable. In particular, if the failed component does not recover, the MA is lost and never returns to its owner [7].

In paper [5] and [7] we propose a group based layered Hierarchical Fault Tolerance Protocol (HFTP) for MA. HFTP can tolerate various kinds of faults that may appear during the life cycle of MA. This paper explores the possibility to adopt HFTP to solve the problem of blocking. If malicious behaviour of host is treated as fault then this fault may be masked by grouping of host and tolerated successfully. Following section introduces the architecture of HFTP and modifications required to solve the MA blocking problem.

II. HIERARCHICAL FAULT TOLERANCE PROTOCOL (HFTP)

HFTP has been proposed to tolerate different kinds of faults that may occur during the life cycle of MA. During its life cycle, a MA can fail due to some uncaught exception, or due to the failure of the MAS, or its components or the host machine. The MA may also be lost on its way or blocked due to link failure. Since failure occurs at different places due to different reasons, specialized approaches have been used to tolerate different kinds of faults. This paper does not discuss the details about the faults and tolerance schemes of HFTP but only introduces the architecture and components of HFTP that can cooperate to solve problem of blocking.

A. System Model

The system model used by HFTP divides the open network like internet into regions. Instead of doing a logical partitioning of the network into regions and then arranging them into hierarchy, it uses the existing technology to serve its purpose. Internet is network of networks. Networks are connected with each other via router [3]. HFTP treats each network as a region and router as the centralized component in each region. Router in proposed architecture is not passive but plays an active role. A MA wishes to visit a host within a network, first arrive at the router. MAS is installed at router but it is responsible only to receive and pass the MA to the designated host in the network, not to execute them. Routers are assumed to be fault-free and trustworthy. In each network there is shared local storage space (LSS), which is accessible by all hosts and assumed to be fault free and trust worthy. Figure-1 shows the basic architecture of the HFTP[4][6].

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 2, February 2014

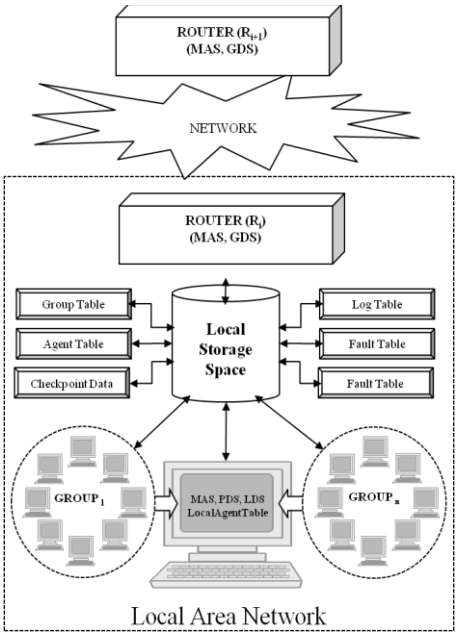

Figure 1: Architecture of Hierarchical Fault Tolerance Protocol

B. Grouping of Hosts

Concept of grouping the hosts to tolerate faults has been successfully used by various researchers working in the field of distributed computing. In each network hosts are grouped based on the kind of services they offer. Hosts are grouped logically. Within each group one host acts as in-charge. A MA submitted at one host for execution may be executed on any member of the group. Group in-charge is responsible for executing a deterministic algorithm to assign a host to the MA submitted to the group impartially. Its algorithm can be modified to distribute the load of a malicious host among the trusted members of group during recovery and protects the MAs were executing on the malicious host. Within a network, there may be various groups. One host may be part of one group only. Each group appears as a single host to other hosts of the distributed system. Hosts in the group communicate each other by using group communication services designed to operate within the LAN.

If a group in-charge fails, another member host of the group takes its place according to a predefined priority. Once a new host becomes the in-charge, it is its responsibility to inform other members about this change.

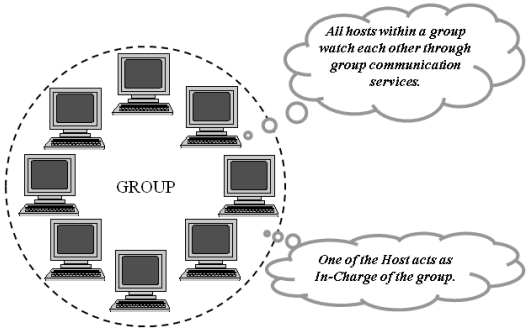

Figure 2: Group within a Network

C. Data Structures for Table

Once a group is formed, records of each group and all agents running on each group of network are put in various tables implemented in LSS.

1) Group table

The *GroupTable* contains information about which host is part of which group and which is the in-charge of the group. Each group is uniquely identified by a *GroupId*. A group is characterized by its in-charge, active members and size. *GroupTable* is accessed by all hosts but modified only by the group in-charge. Entries in the *GroupTable* are modified when a host found malicious or recovers and become trustworthy again. Table-2 shows the structure of *GroupTable*.

TABLE 1: GROUP TABLE

GroupId	In-charge	Size	Member HostId

2) Agent Table

The *AgentTable* stores information about which MA is running on which group as well as on which host. Table-2 shows the structure of *AgentTable*.

TABLE 2: AGENT TABLE

AgentId	GroupId	HostId

3) MalHostTable

This table maintains a list of hosts currently unavailable as their behaviour has been found malicious or they are under recovery. Table-3 shows the entries in *MalHostTable*.

TABLE 3: MALICIOUS HOST TABLE

GroupId	List of Malicious member HostId

All these tables are accessed by all hosts but can be updated only by the group in-charges or the Router. Group and MalHostTable are updated when a host becomes inaccessible or recovers. Similarly AgentTable and GroupTable are updated when a MA is submitted or has completed its execution at a host.

4) Local Agent Table

A *Local Agent Table* is maintained at each host to keep records of all the MAs running on each host of the group it belongs. Its entries are shown in Table 4.

TABLE 4 - LOCAL AGENT TABLE

AgentId	HostId

D. Layered Architecture of HFTP

HFTP uses a 3-level layered architecture. Server at the lowest layer is Personal Daemon Server (PDS), at the middle level Local Daemon Server (LDS) and at the

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 2, February 2014

highest level is Global Daemon Server (GDS). LDS and PDS are installed on each host of the network. GDS is installed on routers only. This section briefly describes the functionality of each server.

1) Personal Daemon Server (PDS)

PDS is the server at the lowest layer, installed on each host of the network. It monitors the MAS and all MAs running at the host by maintaining a thread for each. In case behaviour of MA or MAS is found malicious, it informs other group members about this observation and initiates recovery procedure of MAS. Figure-3 shows its responsibilities in brief.

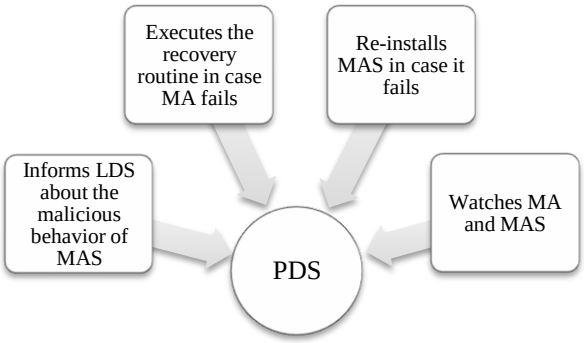

Figure 3: Functions of the Personal Daemon Server

2) Local Daemon Server (LDS)

LDS is the server at the middle level, installed on each host of the Network. It is responsible for detecting the malicious behaviour of the host and for executing all group communication services within the group. In case either a host or MAS installed on the host found malicious, it distributes the load of the malicious host among trusted members of the group. Figure-4 shows its responsibilities in brief.

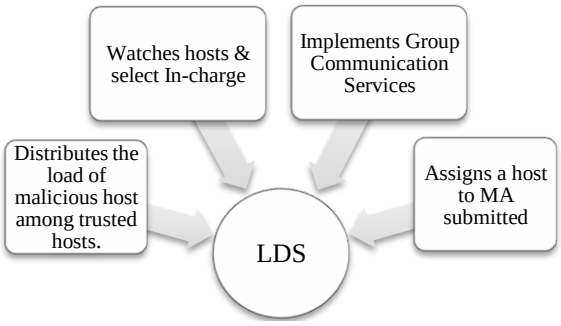

Figure 4: Functions of the Local Daemon Server

3) Global Daemon Server (GDS)

The GDS is the server at the highest level. It is installed only at the Router. MAs arrive and migrated via router. GDS is responsible for implementing various security, location management and fault tolerant routines. GDS is responsible for evaluating the trustworthiness of MAs and hosts in the network. If it found an incoming MA trust worthy, MA is passed to the target group in-charge.Figure-5 shows its responsibilities in brief.

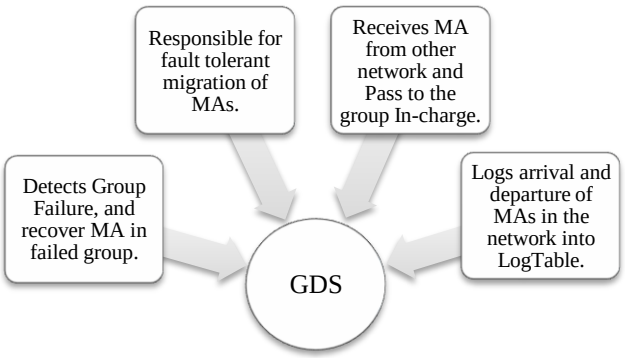

Figure 5: Functions of the Global Daemon Server

E. Protocol

Above sections discuss the various components and their roles in HFTP. As in this paper we are exploring the possibility to modify the HFTP to solve the MA blocking problem, this section explains, how these components can cooperate to mask the malicious host and continue with the execution of MA. As discuss earlier a group appears as a single host to the rest of the system, a user who launch the MA, provides its ordered itinerary as a list of *GroupId*. MA moves from a group in one network to group in other network via router. The steps followed since a MA is arrive at a network are-

- MAS is installed at the Router receives the MA.
- GDS Logs an arrival entry logarrive in *LogTable*.
- Checks the next address in the MA itinerary.
- Passes the MA to the in-charge of the group.
- LDS installed at Group In-charge executes a deterministic algorithm to select a trusted host with minimum load to execute the MA and transfers MA to the selected host.
- Agent Tables are updated accordingly.
- A message is broadcasted among the members to update their *LocalAgentTable*.
- As soon as MA arrived at host, host checkpoint its state and system state in LSS.
- PDS installed at host, starts a new thread to watch the execution of MA.
- MAS installed at host execute the MA and checkpoint its state after every successful transaction or whenever needed.
- After successful execution of MA, it is submitted at the Router.
- PDS terminates its thread after successful migration and deletes useless checkpoint data.
- Local and Global Agent Tables are modified accordingly.

F. Malicious Host and Recovery Steps

As mentioned earlier, it has been assumed that every network implements some Security Management System to detect whether a host is trusted or malicious and maintains a list of failed host. We further assume that some recovery management system is also implemented in the network to clean the malicious host and make it trust worthy again. Security Management System of the network may identify a host malicious any time and inform the group in-charge about its finding. This section

discusses the masking of this malicious host and recovery of MA hosted by it by the group.

- Once the router informs the in-charge about malicious behaviour of host, it distributes the load of malicious host among remaining group members and updates tables accordingly.
- Newly selected hosts recover the MAs from their last checkpoint state and continue their execution.
- Recovery of malicious host is the responsibility of its own network recovery management system.
- If the failed host is the group in-charge, then remaining members of the group cooperatively elect a new group in-charge based on priorities.
- Newly elected in-charge then updates tables accordingly.
- If malicious host was the only host in the group, then group failure is recognized by the router and all the MAs running in the group are blocked until a member become trusted again.
- If the order of the host to be visited is not fixed then MA is migrated to other host and try the failed host latter until at least one of the host become trusted.

III.CONCLUSION

As discussed in the paper, it has been observed that by using the grouping of host MA blocking problem can be successfully solved for the HFTP and continue the execution of MA even if the target host is found malicious. HFTP has already been modeled using the Colored Petri Net (CPN) and verified for its performance [8] based on the simulation results. Since most of the approaches used here are well known and has already been implemented successfully so it is quite reasonable to accept that, this architecture once implemented will solve the concern issues successfully. Its efficiency or comparative performance analysis is possible only after the implementation.

The problem of blocking occurs when itinerary of MA is static and order is fixed. In order to avoid blocking, the user can also specify a list of alternative hosts at each steps of the MA's itinerary. If MA found one host malicious in its itinerary, it may be migrated to an alternative host in the itinerary.

If order is not fixed then in case of blocking MA may migrate to any other trusted host of the itinerary and may try to visit the malicious host after sufficient time till host has been cleaned at become trustworthy again.
For some applications, the itinerary of MA is determined dynamically by the MA itself. In this case, if MA is blocked it finds some alternative host that offer the same services and migrate to that host. If required it may visit the malicious host again when it has been cleaned and trust worthy again.

REFERENCES

[1] De Capitani di Vimercati, S.Foresti, S.Jajodia et al. Integrating Trust Management and Access Control in Data Intensive Web Applications,ACM Transactions on the Web (TWEB) 6,2, 1-44 (2012).

[2] Habib S., Ries S., Muhlhauser, M., Towards a Trust Management System for Cloud Computing, In Proc. of IEEE 10th Int. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom'11). Changsha, China (2011)

[3] Patel, R.B. 2004. Design and implementation of a secure mobile agent platform for distributed computing', PhD Thesis, Department of Electronics and Computer Engineering, IIT Roorkee, India, Aug.

[4] Pathak H., A Novel Hybrid Security Architecture (HSA) to provide security to Mobile Agents and the Executing Host, Proceedings of the International Conference on Communication, Computing & Security Pages 499-502, Rourkela, 2011.

[5] Pathak H., K. Garg, Nipur, "CPN model for Hierarchical Fault Tolerance Protocol for Mobile Agent Systems", in proceedings 2008 International Conference of Networks (ICON 2008), New Delhi, India, December 2008.

[6] Pathak H., K. Garg, Nipur, "Design, Validation, Simulation and Parametric Evaluation of a Fault Tolerant Network Trading System Using Mobile Agent" in Journal of Information and Operations Management (JIOM), Vol 3, Issue 1. Feb 2012.

[7] Pathak H., K. Garg, Nipur, "Three Layered Hierarchical Fault Tolerance Protocol for Mobile Agent System" in International Journal of Scientific and Engineering Research (IJSER), Vol. 2 Issue 1 (2011).

[8] Pathak H., K. Garg, Nipur, "Performance Analysis of Hierarchical Fault Tolerance Protocol for Mobile Agent Systems", in Journal of Computer Science (JCS), Vol.5,issue 2, pp 118-124 (2011).