A New Hierarchical Fault Management System (HFMS) of Mobile-Multi Agents

Mrs. Swati Singhal[1]
Department of Computer Science,
Gurukul Kangri Vishwavidyalaya,
Haridwar
India

Dr. Heman Pathak[2]
Department of Computer Science,
Kanya Gurukul Campus,
Dehradun,
India

***Abstract*—Mobile Agent technology has become a new paradigm for distributed real-time systems because of their inherent advantages. In any distributed system, along with other issues, survivability and fault tolerance are vital issues for deploying Mobile Agent for real applications. All Mobile Agent based applications face reliability problems due to the failure of agent platform, host and communication link etc. Mobile Agent technology has drawn attention of various researchers working in the area of distributed computing. Researchers have realized the benefits of using Mobile Agents over other existing technology and explore various area of application for Mobile Agents such as Electronic Commerce, Network Management, and Distributed Applications etc. However to fully deploy Mobile Agents in practice a number of challenging issues especially security, fault tolerance and privacy need to be addressed. The scope of this paper is limited to address fault tolerance problem of Mobile Agents in a multi agent environment. This paper identifies various faults in life cycle of Mobile Agent and proposes a comprehensive solution to tolerate all these faults. Paper also identifies the situations when Mobile Agent gets blocked and suggests several ways to avoid these blocks and continues its execution. Proposed solutions combine various existing fault tolerant mechanism to tolerate different kinds of faults. Proposed approach is hierarchical in nature, which combines both centralized and distributed approaches to take advantages of both. Thread base mechanism has been used to detect the failure of Mobile Agent and its executing environment. Heart-beat message passing and Ping command based techniques have been suggested to detect failure of Agent servers. Mobile agent system places have been introduced in this paper for Mobile agent execution. Modified itinerary options have also been explored for tolerating faults and to avoid blocking. Rear guard and acknowledgement based mechanism has been used to tolerate link failure and to ensure fault free migration of Mobile Agents across the networks.**

Keywords-Agent (MA), Mobile Agent System (MAS), Multi Agent System (MUS), Fault Tolerance, Ping Method.

I. INTRODUCTION

Mobile Agent (MA) technology has emerged as major programming paradigm for distributed applications. MAs are programs which are dispatched from a source computer and run among a set of networked servers until they are able to accomplish their task. A prerequisite for their use, however, is that they should be executed reliably independent of failures. Improving the survivability of MAs in presence of various failures is an important issue in order to guarantee continuous execution of MAs. Thus it is very important to make MAs fault tolerant. This paper proposes a novel hierarchical approach that handles the faults very effectively in Multi Agent environment. Various researchers working in the area of MA technology have proposed various mechanisms to tolerate different kinds of faults in MA life cycle. [1]

Various research papers are available where an attempt has been made to identify various parameters and compare and analyse the existing approaches against these parameters. Although we have also explored and analysed the various existing fault tolerance approaches but our comprehensive literature survey is not presented in this paper. Instead we are only summarizing the various approaches and related references here.

II. POSSIBLE FAULTS IN THE LIFE CYCLE OF MA

MAs react dynamically and autonomously to the changes in their environment, which makes them robust, and fault tolerant. They have the ability to distribute themselves in the network in such a way as to maintain the optimal configuration for solving the particular problem. If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch themselves and continue their operation on another host in the network. In spite of being proactive, intelligent and dynamic there are various faults present in the executing environment of MAs that may cause premature termination of MAs. Consider an application where a MA visits diverse hosts distributed across many networks and connected by a network backbone. In this distributed system, there are a number of possible failures that can interrupt the execution of MAs. Hardware failures with individual machines or with the underlying network can result in loss of the agent. Software failures can cause the premature termination of the agent and may depart the system in an unpredictable state. An unanticipated failure at any of these points causes the entire application to fail. In this section we detail the various faults that can occur during the life cycle of MA.

A. Failure of Mobile Agents

The MA is a piece of code that carries with it data, code and execution state. During its execution, the MA can fail due to faulty computation, uncaught exceptions, runtime errors or it may be interrupted by system components or faulty processes. Due to any of these reasons, the MA can't carry on its execution.

B. Failure of Components of MAS

The Mobile Agent System (MAS) provides the infrastructure to create, host, execute and migration of the MA. It is a piece of code consisting of various components. Any of its components may fail due to overload, resource unavailability, faulty communication units, incomplete agent directory [2], uncaught exceptions, runtime errors etc. This failure can be divided in to two parts. One is place failure and second is MAS failure. Every time when a MA need to execute on a host platform, MAS on the host provides a separate place to execute MA. A place failure cannot block the MA execution but in case of crashed MAS, It causes the crash of all agents currently running or hosted on it.

C. Failure of Host Machine (HM)

The breakdown of a HM causes MAS and all MA running on this machine to crash. Failure of a host to which an agent intends to visit will force the agent to block. Host breakdown can be temporary or permanent. Recovery of the HM is the liability of the network manager to which it belongs.

D. Communication Link Failure

Apart from infrastructure failure, the communication link may also go down at any time. Due to link failure, a MA may be lost on its way. Link failure may stop the movement of an agent from one host to the next and the agent may get blocked. Link failure can also lead to isolation of a single host. Network partitioning is an unfortunate result of such a communication link failure. The agent gets trapped inside one of these partitions. If all the unvisited hosts and the agent are in the same partition, the agent can still complete its itinerary. However, if the agent and the hosts to be visited are in dissimilar partitions, it is impracticable for it to complete its itinerary until some of the links are recovered.

E. Other Faults

During the lifecycle of a MA, a variety of additional faults may take place at different stages, which must be caught and handled at the Agent, System or Host level [3]. One of these faults is the problem of a faulty processor. Executing the MA on a faulty processor might corrupt the agent's program and/or data, resulting in the MA deviating from its normal behavior. In this paper proposed protocol works only on the infrastructure and link failures only.

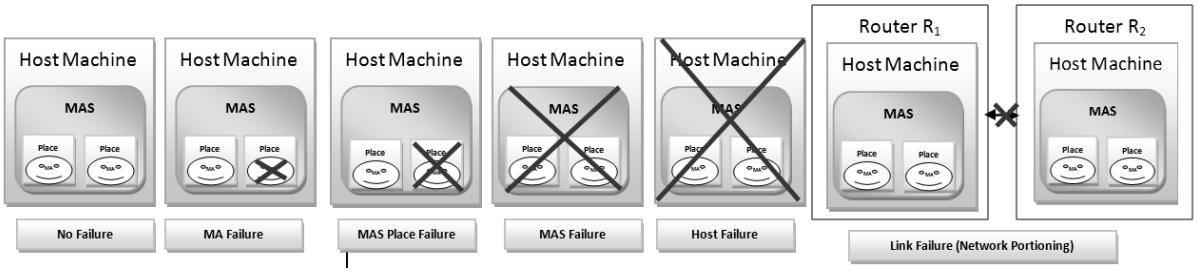

Figure 1. Different Fail Cases

III. FAILURE AND THE BLOCKING PROBLEM

In some cases, if MA migrates from one host to other host and the target host is not available or nor ready to receive the MA, MA may get lost. MA also may get lost on its way if any of the links from source to target host get failed. In order to avoid the loss of MA due to unavailability of target host or link failure, this paper proposes a precautionary measure in which before MA moves from one host to other host in its itinerary, the target host is first checked whether it is available/ready to receive the MA or not. To check the availability of target host Ping command may be used or other specialized application program may be developed. This paper proposes to use simple Ping command. Ping is a basic Internet program that allows a user to confirm that a particular IP address exists and can admit requests [4][5]. MA is migrated to target hosts only if it is available to receive it, otherwise MA gets blocked until target host becomes available. AMA execution is called blocking, if a particular unavailability or failure provides steps forward in the MA execution unfeasible until the unavailable component (e.g., Host Machine) becomes available. Blocking MA executions are undesirable. In particular, if the failed component does not recover, the agent is blocked and waits for the unavailable host. Following sections proposes some of the solutions to avoid blocking where MA is waiting for failed components. Proposed solutions are based on MA itinerary and its order.

IV. PROPOSED SOLUTIONS TO AVOID BLOCKING

The set of hosts to be visited by a MA during its life cycle is called its itinerary. Generally a MA moves from one place host to another host according to its itinerary. Itinerary of MA can be static or dynamic. In case of static itinerary, it must be initialized at the time of MA is created. A static itinerary is entirely defined at the MA source and does not change during its execution. A dynamic itinerary of the MA is determined dynamically by the agent itself. However, at least the first host in the itinerary should be known a priori. The sequence of hosts visited by MA in its itinerary is its order. Order may also be determined statically or dynamically. Static order means hosts in the itinerary must be visited in the order specified by the creator of MA while dynamic order is flexible where specified hosts may be visited in any order.

A. Case 1: Static Itinerary Static Order (SISO)

For some applications, the itinerary is static and the order in which MA completes its itinerary is fixed. Since the order of visit is fixed statically, in this method, MA can continue its execution on the target hosts (in its itinerary) serially (one by one). In case of blocking, MA continuously waits for availability of target host.

To avoid the blocking of MA for SISO case, we propose that the creator of MA while providing the itinerary of MA, then instead of providing the sequence of host addresses to be visited, creator must also provide alternative host addresses at each stage that may be visited in case one host is unavailable or failed.

Original Itinerary - [Ha1, Ha2,…Hai, … Han]
Modified Itinerary - [(Ha11, Ha12,... Ha1j), (Ha21,…Ha2k), …(Han1,.... Hanm)].

At any stage of MA lifecycle, if target host is unavailable/ failed then instead of blocking MA, components of MAS responsible for migration of MA may choose an alternative host from the list provided by the creator and selected for migration. In this way blocking could be avoided and MA may continue its itinerary. Many replication based approaches and some group based approaches already use this concept where at each stage MA may be executed on one of the several possible target hosts offering same kind of services. [4, 7, 8, 9, 10 and 17].

B. Case 2: Static Itinerary Dynamic Order (SIDO)

For some applications, the itinerary is static, but the order in which MA completes its itinerary is not fixed and MA can visit the hosts in the itinerary in any order. MA starts its execution from the first host in this itinerary (Serially) but in case when target host is not available/ failed, instead of get blocked and waits for failed components to recover, we propose that MA can be migrated to other available hosts in its itinerary and can try the failed host later when it becomes available. Proposed approach avoids blocking by changing the order of the itinerary but in case of permanent host failure MA may get blocked. We propose if failed component does not recovered within some pre specified time, it is assumed to be failed and MAS may send MA to an alternative host offering same service and continue MA execution.

C. Case 3: Dynamic Itinerary Dynamic Order (DIDO)

For some applications, the itinerary is determined dynamically by the agent itself. Here MA determines its next host to be visited by itself. If the target host is not available, MA can dynamically find another host and can continue its execution without blocking.

V. HIERARCHICAL FAULT MANAGEMENT SYSTEM (HFMS)

In the proposed Hierarchical Fault Management System (HFMS) an attempt has been made to provide complete solutions to tolerate all kinds of faults discussed earlier. HFMS combines both centralized and distributed approaches and takes advantages of both. The system model used by Hierarchical Fault Management System (HFMS) 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 infrastructure to serve its purpose. Internet is network of networks. Networks are connected with each other via router. HFMS 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 arrives at the router. MAS is installed at router and all hosts of the network. It provides the basic infrastructure required to support creation, migration and execution of MAs. Routers are assumed to be fault-free and trustworthy. In each network there is shared local storage space (LSDS), which is accessible by all hosts and assumed to be fault free and trust worthy. [3, 11 and 15]

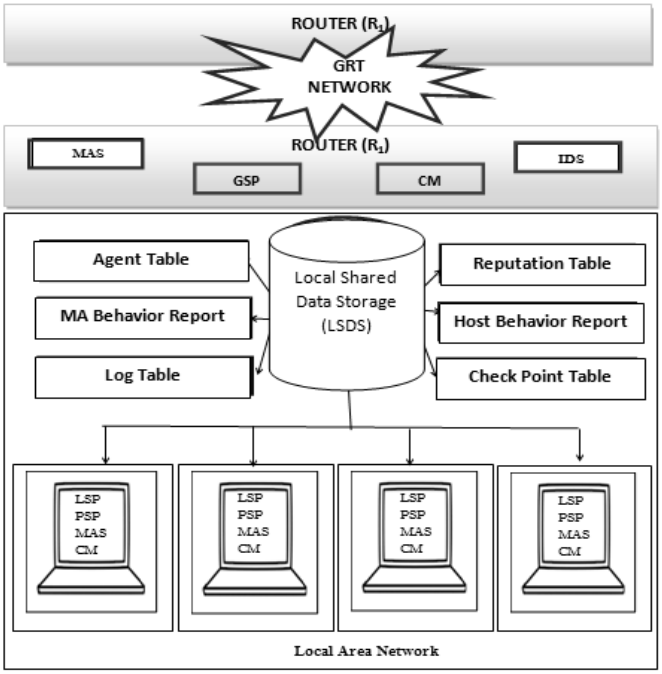

Figure 2. Hierarchical Fault Management System

A. System Components

System model used by the HFMS is consisting of three main hardware components Router, Hosts and Local Shared Data Storage (LSDS). Various software components installed on each are explained in the following section.

1) Router

Routing is the process of forwarding MA from one network to another but in the proposed system, router plays key role to detect and tolerate various faults and insure fault free migration of MA from one router to other. Components installed on the router are listed here.

a) Global Service Provider(GSP)

The GSP is the server at the highest level. It is installed only at the Router. It is responsible to receive every incoming MA and then passing it to the appropriate host of its own network. It also receives MAs from its own networks that are ready to migrate to hosts in other parts of the network. GSP is responsible for performing all functions required for fault tolerant migration of MAs in the local and global network by implementing the Agent Transfer Protocol.

b) Checkpoint Manager (CM)

Every time when a MA enters in a network, CM is responsible to save the MA and its execution state to LSDS periodically and after every successful transaction. This checkpoint data can be used to recover the MA lost by failure of MAS or its executing host.

2) Host

Host is a computer in the network which offers services to the MA. It provides executing platform to the MA. Following section discuss the components installed at each Host.

a) Local Service Provider (LSP)

LSP is the server at the middle layer, installed on each host of the network. It is responsible to communicate with GSP for sending acknowledgement and informing about MAS failure. It is also responsible to update Agent Table, initiate recovery procedure in case of failure of MA and MAS are detected.

b) Personal Service Provider (PSP)

PSP is the server at the lowest layer, installed on each host of the network. It monitors the MAS as well as all the agents running at the host. It watches the agent as well as the MAS by maintaining a thread for each. In case the MA fails or MAS crashes due to any reason, it informs its LSP about the failure.

c) Checkpoint Manager (CM)

This is responsible to save the MA and its execution state to LSDS periodically and after every successful transaction. This checkpoint data can be used to recover the MA lost by failure of MAS or its executing host.

3) Local Shared Data Storage(LSDS)

Each network is assumed to maintain a fault free storage space. This space is accessible by all hosts and components installed at router. It is used to store various tables and mailbox of executing MAS. Information stored in these tables is used to perform recovery from the failed states. Following sections describes the tables and their entries-

a) Log Table

A *Log Table* is maintained in LSDS to log the arrival of every MA from other parts of network. The same table may be used to log departures for every MA migrating to other parts of network. These entries are used to detect a fault during migration.

b) Agent Table

The *Agent Table* stores details of, which agent is running on which host. New entry is added when MA is submitted to a host for execution. Entry is updated or deleted when MA is transferred to other host or migrated from the network respectively.

c) Failed Host Table

This table maintains a list of hosts currently unavailable to host and executes a MA either due to failure of MAS or hosts themselves.

d) Checkpoint Table

This table maintains an entry of mobile agent id with its check point state.

VI. STEPS OF MA EXECUTION IN A FAULT FREE ENVIRONMENT

A mobile agent is a particular type of agent with the ability during execution to migrate from one host to another where it can resume its execution. Computer hosts, or platform, provide agents with environments in which they can execute. This environment should be secure and fault free. MA execution can be successful only in a secure and fault free environment. The following points show the steps of MA execution in a fault free environment.

- A MA wishes to be executed on a host of the network arrives at Router R_i(say)
- GSP at R_i receives the MA and log an arrival entry in LogTable.
- It creates the clone of MA and saves it in Checkpoint Table.
- It then transfers the MA to the Target Host.
- MAS installed at Host receive the MA.
- LSP installed at Host sends an acknowledgement to GSP and adds or update an entry in Agent Table (AgentId, target HostId).
- PSP installed at host starts a thread to watch MA. PSP also maintains a thread to watch MAS.
- MA execution starts at host, after every successful execution step, MA's state is check-pointed in CT and clone is updated.
- After successful execution of MA at host, MA is transferred to GSP at R_i and Agent Table is updated.

- Thread started by PSP terminated automatically with the execution of MA.
- GSP first creates the clone of MA and saves in CT and then checks the address of next host in the itinerary of MA.
- If next host is in the same part of network, GSP transfers the MA to target Host.
- If next host is in the other part of network then GSP performs following steps-
    - Delete the entry form the Agent Table
    - Log a departure entry in the Log Table
    - Transfer the MA to the target Host network Router R_{i+1}.
- MA keeps moving this way until it completes its itinerary.

VII. FAILURE CASES AND RECOVERY PROCEDURE

During the complete life cycle of a MA, it may fail at any stage due to different reason. In the following section, we give the various faults that may occur during the life cycle of a MA and the schemes used by HFMS to tolerate them.

A. Mobile Agent Failure

A MA is a piece of code which itself may fail during its execution due to some programming/data error or some uncaught exception or run time failure. Failure of a MA can be avoided by providing the exception handling code for all known cases and minimizing the programming error. [12, 13]

But still if a MA fails, its failure is recognized by the PSP, who is watching it. PSP informs LSP about this failure. Since MA cannot be recovered and continue its execution until a recovery routine is provided by the user, who has launched the MA. [5] LSP performs the following recovery routine.

- Rollback all uncommitted transactions and bring the system at a consistent state from checkpoint data, as MA failure may leave system in an inconsistent state.
- Prepare a failure report containing the details such as date and time of fault, IP address of host, line number of code and other descriptions of fault and request to its MAS to create a new Agent say Recovery Agent (RA) to carry the failure report and partial result to the user. RA then transferred to GSP at router to be forwarded to Base Host (BH) creator of MA.

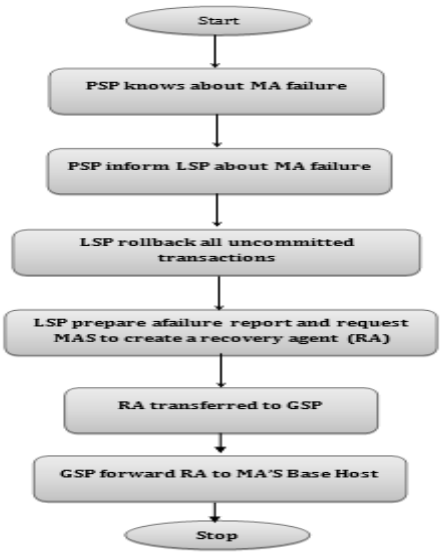

Figure 3. MA Failure

B. Mobile Agent System Failure

MAS is a software program which may also fail/crash due to overload, resource unavailability, uncaught exception, run time error or some other reasons. Due to failure of MAS, all MAs running on it will get lost. This failure can be divided in to two parts. One is place failure and second is MAS failure. Every time when a MA need to execute on a host platform, MAS on the host provides a separate place to execute MA. A place failure

cannot block the MA execution but in case of crashed MAS, It causes the crash of all agents currently running or hosted on it.

1) MAS Place failure

In order to tolerate a MAS place failure we propose that every time when a MA need to execute on a host platform, MAS on the host provides a separate place to execute MA. If this place fails then MAS takes the following actions.

- MAS create another place for MA execution.
- MAS concerns check point table from LSDS and get its check point state.
- MAS resume MA execution on the created new place. Hence A place failure cannot block the MA execution.

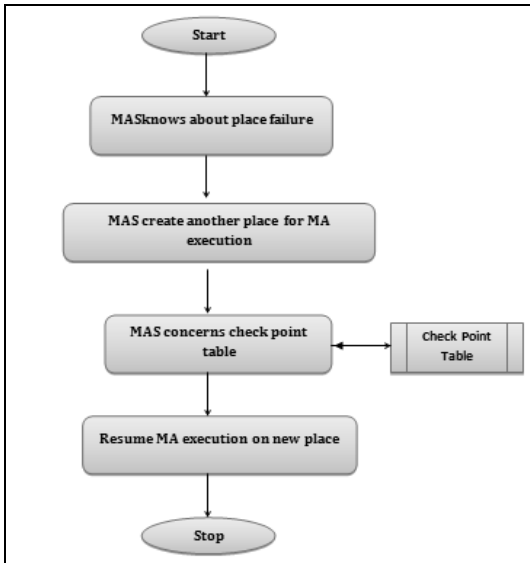

Figure 4. MAS Place Failure

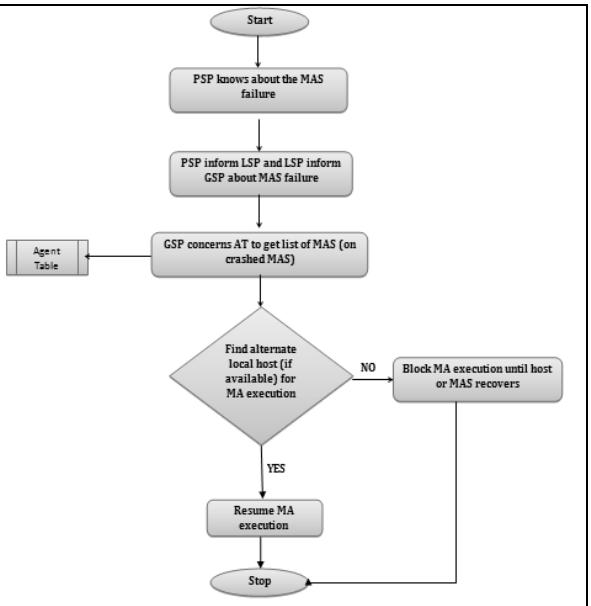

Figure 5. MAS/Host Failure

2) MAS Failure

In order to tolerate a MAS/Host failure we propose that user must provide a list of Host at each stage of execution of MA if its itinerary is static otherwise MA will be blocked till the failed host or crashed MAS recovered. If itinerary is dynamic MA is assumed to generate a list of alternative hosts at each stage of execution. [11, 12 and 13]

PSP installed at the host watches MAS by maintaining a thread for it. If MAS crashes, its failure is immediately recognized by the PSP.PSP informs the LSP and LSP informs GSP about this failure. LSP immediately starts the recovery of MAS which may take some time but lost MAs can't be recovered. GSP then perform following recovery steps-

- GSP first concern the Agent Table to get the list of MAs were executing on the crashed Host (MAS).
- One by one it then checks for each lost MAs, if user has provided any alternative Host List for this stage of MA or not.
- GSP then selects a new Host from the alternative Host List for MA execution. A preference is given to a local host.
- If alternative local host is available, updated saved clone of MA is transferred to new host, where it resumes the execution of MA. Agent Table is updated accordingly.
- If new selected host is in other part of the network, updated clone is transferred to the target router and various tables are updated accordingly.
- If no alternative host is provided MA will be blocked till the failed host is recovered.
- At the target host MA resumes its execution.
- Same procedure is repeated for all lost MAs.

C. Host Failure

A host is a machine in a network that provides a logical execution environment to MAs and executes them. Hosts may go down at any time; consequently MAS as well as all the MAs running on that system will be lost.

In order to tolerate host failure various researchers has proposed replication based techniques [14] where at each stage of execution of MA, it is transferred not at one host but at group of hosts and one of the Host is chosen to execute the MA while others are watching it. If the executing host fails, other hosts from the group are given responsibility to continue execution of MA. Replication based approaches are not only consume important resources of the network such as host's CPU cycle time, network bandwidth etc. but it is also difficult to enforce exactly once execution of MA. Since Host failure is not a frequent event so, instead of replication based approach we propose the heartbeat message technique to detect the host failure. [15, 16] Following steps are followed to detect host failure-

- Each host of the network periodically sends a heartbeat message "I am alive" (say) to GSP.
- In case GSP does not receive a message from a host, it suspects the host failure.
- Since message may not be received due to link failure, GSP confirms the host failure by using Ping command.
- In case GSP detects a host failure, it adds this host to the list of failed host and initiates the recovery of the host.
- GSP then starts the recovery of MAs that were hosted by the failed host. Recovery of lost MAs steps are same as in case of crashed MAS, so not repeated here.
- If links failure is detected, GSP initiated link recovery routine. Every network has its own network fault management system to recover the failed host and links within the network, so these routines are not discussed in the paper.

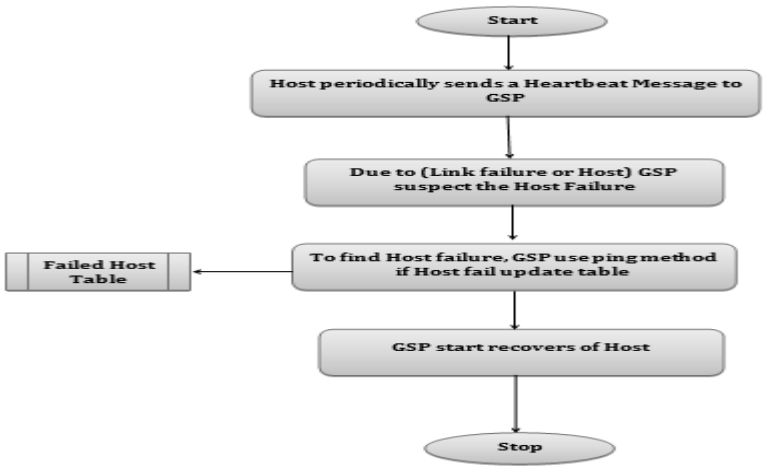

Figure 6. Host Failure

D. Communication Link Failure

All the hosts within the network as well as all the networks are connected with each other through communication links. These links are not fault free and may fail any time. If communication link fails during the migration of MA, then MA will be lost in its way. We propose an acknowledgement based Agent Transfer Protocol (ATP) for both local and global migration of the MA.

1) Agent Transfer Protocol (ATP) for Local Migration

GSP installed at router is responsible for transferring MA received to a Host of the network. [19] Following sequence of steps are executed-

Before starting the transfer of MA, GSP first checks the availability of the Host as discussed earlier.

- GSP then spawn a Local Monitor Agent (LMA) at router and transfers the MA to target.
- LMA start waiting for acknowledgement from the Host.
- When MA arrives at Host, LSP sends an acknowledgement to the router.
- On receiving the acknowledgement LMA terminates.

a) Faults Detection and Recovery

LMA will not receive acknowledgement if either MA or Acknowledgement lost on the way due to link failure. In both cases LMA first checks the Agent Table, if there is an entry by host then it assumes that acknowledgement has been lost and terminates. If there is no entry in Agent Table for MA, then it resends the copy of MA to the host and waits for acknowledgement. Here it has been assumed that no link in local network

fails permanently. Link failure is temporary and failed links are recovered automatically by the network management system.

2) Agent Transfer Protocol (ATP) for Global Migration

Concept of this protocol is inspired by the approach of [18], [6] and [13]. In ATP for global migration, GSP first spawns a Global Monitor Agent (GMA) before migrating to other router. Assume that currently MA is at the router R_i and ready to migrate to router R_{i+1} then following sequence of steps are performed-

- GSP first spawn a Global Monitor Agent (GMA) at R_i.
- MA then transfers to R_{i+1}.
- On arrival of MA at R_{i+1} its GSP first log an arrival entry in log table.
- It then sends an acknowledgement Ack to R_i.
- GMA at R_i is waiting for acknowledgement and if there is no fault, it will receive the acknowledgement on time and terminates.

a) Failure detection and recovery

If GMA at R_i fails to receive acknowledgement on time, It suspects either MA/Ack has been lost or arrived after time out. Once fault is suspected by the GMA, it takes following actions-

- Creates a Probe Agent (PA), which travels to R_{i+1}.
- PA checks the Log Table, if it finds a log entry in the log table of R_{i+1} means Ack has been lost, it send Ack to R_i and terminates.
- If no entry in Log Table for MA is found, it means MA has been lost on its way to R_{i+1}, PA then sends a request to GMA of R_i to resend MA and terminates.
- After sending the PA, GMA again waits for acknowledgement/request for MA.
- If again GMA does not receive acknowledgement or request message from PA, it assumes the network fault and waits for network to resume and keeps sending PA until receives the acknowledgement.

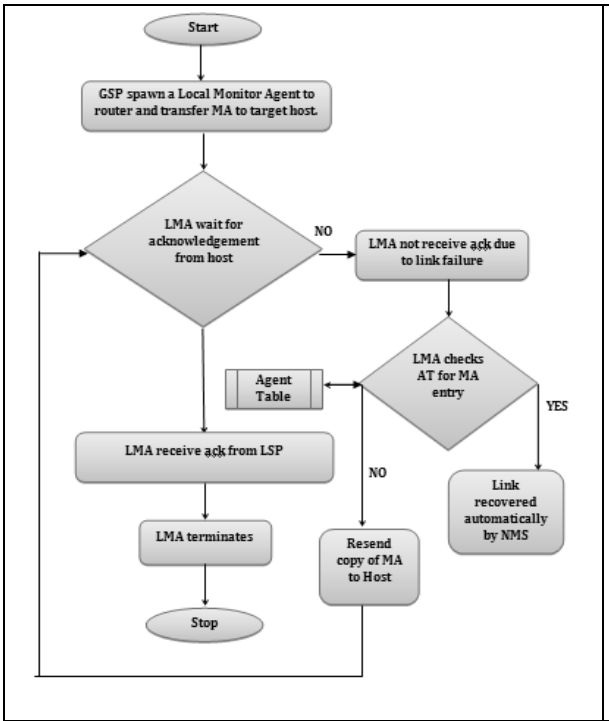

Figure 7. Local Migration

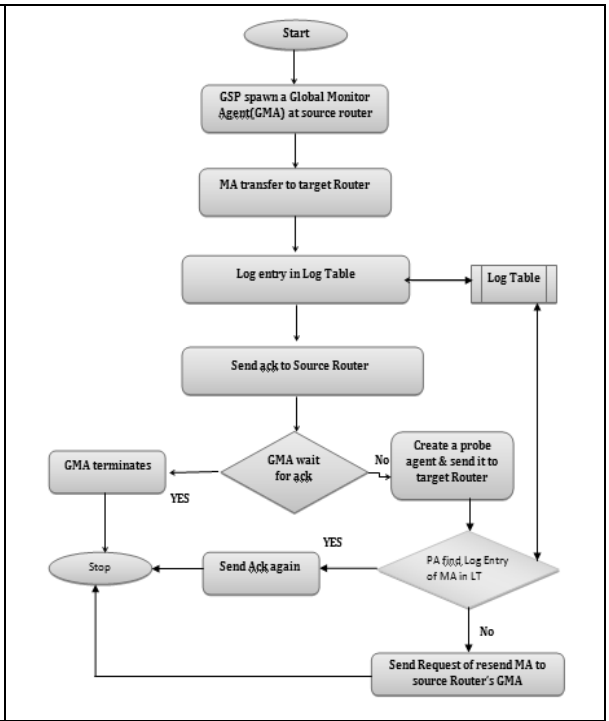

Figure 8. Global Migration

Network partitioning (NP) is an important aspect of the fault tolerance but most of the suggested protocols either does not tolerate Network Partitioning or do so in a very poor manner. The Exactly Once [10] protocol can tolerate any local network partitioning provided it does not affect the communication between nodes in the corresponding domains. It achieves NP by a voting protocol, which ensures that only one worker commits in each domain. The NAP protocol [20] does not tolerate NP but claims that improved environment may tolerate NP.

Authors in [4] and [7] claim to tolerate NP to some extent. For some other protocols like Rollback Recovery [5] some extension to the protocol may achieve NP [18, 21 and 22]. NP is a very rare operation because one network is connected by multiple links from other networks so in any case when one link is breakdown; an alternative path is available between two networks.

VIII. CONCLUSIONS

Generally a support is needed for MA fault tolerance, for guaranteeing transactional behavior of MA applications and for providing reliable forms of MA's coordination. We analyze different approaches of fault tolerance introduced by different researcher and try to give any conclusion about the need of fault tolerance mechanisms. As we have seen different approaches which have their advantage and drawbacks so still there is a need for the good and reliable protocol for MA in terms of fault tolerance. Paper presents Hierarchical Fault Management System (HFMS) of fault Tolerance which uses host combination in MA's itinerary in place of one single host according to their service level and blocking of MA handles in case of host unavailability. Here GSP place an important role for host failure and host unavailability. Before moving of MA to target host, GSP pings target host to check whether it is available or not. Suppose a MA is executing on a host and that host fail, in this case GSP provides another (local or global) host from MA's itinerary. Hence this way a successful and reliable hierarchical based approach introduced in this paper which is used for the execution of MA and provides a fault free environment to it. Modeling of HFMS is our future work through CPN Tool.

IX. REFERENCES

[1] Harvendra Kumar,and A. K. Verma," Comparative Study of Distributed Computing Paradigms", research paper published in BVICAM's International Journal of Information Technology Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

[2] H. Pals, S. Petri, and C. Grewe, "FANTOMAS:Fault Tolerance for Mobile Agents in Clusters", Proceedings International Parallel and Distributed Processing Symposium (IPDPS), 2000, Worksoft J.D.P. Rollim (ed.) pp. 1236-1247, 2002.

[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] Michael R. Lyu, T. Y. Wong, "A Progressive Fault Tolerant Mechanism in Mobile Agent Systems",Proc. Of the 7th world Multi-conference on Systematics, Cybernetics and Informatics, Vol. IX, Orlando, Florida, July 2003, P.P. 299-306

[5] E. Gendelman, L. F. Bic, and M. B. Dilllencourt, "An Application transparent, platform independent approach to rollback-recovery for mobile agent systems", 20th IEEE International Conference on Distributed Computing Systems, Taipei, Taiwan, pp. 564-71, April 2000.

[6] Michael R. Lyu, Xinyu Chen, T. Y. Wong, "Design and Evaluation of a Fault-Tolerant mobile-Agent System", IEEE intelligent System, September/October 2004, 1541-1672/04 pp32-38.

[7] S. Mishra, "Agent Fault Tolerance Using Group Communication", Proceedings of the 2001 International Conference on Parallel & Distri-buted processing Techniques and Application Las Vegas, N V. June 2001.

[8] H. Pals, S. Petri, and C. Grewe, "FANTOMAS:Fault Tolerance for Mo-bile Agents in Clusters", Proceedings International Parallel and Distributed Processing Symposium (IPDPS), 2000, pp. 1236-1247, 2002.

[9] S. Pleisch, "Fault-Tolerant and Transactional Mobile Agent Execution",Ph.D thesis, Ecole Polytechnique Federal of Lausanne, Faculty of In-formation & Communications, 2002.

[10] K. Rothermel, M. Straßer, "A protocol for providing the exactly-once property of mobile agents", In Proceedings of the 17th IEEE Sympo-sium on Reliable Distributed Systems (SRDS), Purdue University, West Lafayette, Indiana, USA, pp.100-108, October 1998.

[11] 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.

[12] 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.

[13] Heman Pathak, Kumkum Gard, Nipr, "Three Layered Hierarchical Fault Tolerance Protocol for Mobile Agent System" in the International Journal of Scientific & Engineering Research (IJSER), ISSN 2229-5518, Volume 2, Issue 1, January 2011.

[14] Zahia Guessoum , Nora Faci , Jean-Pierre Briot ,"Adaptive Replication of Large-Scale Multi-Agent Systems – Towards a Fault-Tolerant Multi-Agent Platform

[15] P. Marikkannu, J.J. Adri Jovin and T. Purusothanman," Fault-Tolerant Adaptive mobile agent system using dynamic role based access control" research paper published in International journal of computer applications (0975-8887) in volume 20-No. 2, April 2011.

[16] Matias Cogorno, Javier Rey, Sergio Nesmachnow," Fault tolerance in Hadoop MapReduce implementation", research paper published in 2013.

[17] Suzanne Sweiti and Amal Al Dweik,"Integrated Replication-Checkpoint Fault Tolerance Approach of Mobile Agents "IRCFT" ,research paper published in the International Arab Journal of Information Technology, Vol. 13, No. 1A, 2016.

[18] D. Johansen, K. Marzullo, F. B. Schneider, K. Jacobsen, and D. Zagorodnov, "NAP:Practical fault-tolerance for itinerant computations", In Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS'99), Austin, Texas, USA, pp. 180-189, June 1999

[19] Rajdeep Bhanot and Rahul Hans," A Secure and Fault Tolerant Platform for Mobile Agent Systems", research paper published in International Journal of Security and Its Applications Vol. 9, No. 5 (2015), pp. 85-94 http://dx.doi.org/10.14257/ijsia.2015.9.5.09.

[20] D. Johansen, K. Marzullo, F. B. Schneider, K. Jacobsen, and D. Zagorodnov, "NAP:Practical fault-tolerance for itinerant computations", In Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS'99), Austin, Texas, USA, pp. 180-189, June 1999.

[21] Hodjatollah hamidi, Abbas Vafaei And Seyed Amirhassan Monadjemi," Evaluation And Checkpointing Of Fault Tolerant Mobile Agents Execution In Distributed Systems ",journal of networks, vol. 5, no. 7, july 2010. Pages 800-807.

[22] R. B. Patel, K. Garg, "PMADE – A Platform for mobile Agent Distribution & Execution", in proceedings of 5th World Multi Conference on Systemics, Cybernetics and Informatics (SCI2001), Orlando, Florida, USA, July 2001, Vol. IV, pp. 287-292.

[23] Heman Pathak, Swati Aggarwal," Performance Analysis of Hierarchical Location Management Scheme to Locate mobile Agents", International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 3, March 2016, ISSN (Online) 2278-1021ISSN.