ISSN 2229-5518

Three Layered Hierarchical Fault Tolerance Protocol for Mobile Agent System

Heman Pathak, Kumkum Garg, Nipur

Abstract— A Mobile Agent (MA) is autonomous and identifiable software process that travel through a network of heterogeneous machine and act autonomously on behalf of user. Improving the survivability of MA in presence of various faults is the major issue concerns with implementation 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 should be used to tolerate different kinds of faults. This paper presents a brief introduction of Hierarchical Fault Tolerance Protocol (HFTP) for Mobile Agents. The proposed protocol is hierarchical in nature, which works at three levels. Based on the experience gained from prior work, our approach is to use the concept of fault masking without replication at one level. This ensures that failure is not visible to the MA through grouping of hosts within a network. At another level rear guard based fault detection and recovery based approach has been used. A thread based approach has been used to detect faults at the lowest level. In this way, the protocol tolerates various kinds of faults and takes the advantage of both centralized and distributed approaches. HFTP can tolerate host failure, system failure as well as link failure by grouping the hosts within a network and rear guard based migration of MA in the global network. A well known modeling tool Color Petri Net (CPN) has been used for architectural model of HFTP. Simulation results have been used to check the performance of HFTP in presence of various faults.

Index Terms—Mobile Agents, Fault Tolerance, Colored PetriNets, Mobile Agent Systems

1 Introduction

MOBILE Agent (MA) [1], [16] is an emerging technology that is becoming increasingly popular. Although potential usefulness of the MA computing paradigm has been widely accepted, MA technology has not yet found its way into today's more prominent applications. Before MA applications begin to appear on a large scale, Mobile Agent System (MAS) needs to provide infrastructure services to facilitate MA development. Among these are security, management of MA, fault tolerance, and transaction support. In this paper we are introducing the Hierarchical Fault Tolerance Protocol (HFTP) for MAs and its Colored Petri Net Model. CPN model has been used to analyze and check the performance of HFTP in presence of various faults.

2 Hierarchical Fault Tolerance Protocol

During the life cycle of MA, it can fail due to some uncaught exception or due to the failure of MAS or its components or host machine. 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 fault tolerance approaches should be used to tolerate different kinds of faults. Suggested protocol is a hierarchical protocol, which works at three levels. Based on the experienced gained from prior work, this approach has been designed to use fault masking by grouping hosts within a network at one level while fault detection and recovery by using rear guards at another. For fault masking replication is the most common technique but it requires to use multiple host to implement one logical host and also enforcing exactly once execution of mobile agent is critical in replication based approaches [10], [16], [17]. Proposed protocol achieves fault masking through grouping of hosts [6], [9] within the network. Since it is not using the replication so there is no need to enforce exactly one. There is always one active copy of the MA in network.

2.1 Assumptions and Requirements

Some of the assumptions / requirements for HFTP are –

- All networks are connected via Routers.
- Routers are fault free i.e. MA can not be lost there, but may be blocked due to link failure.
- There is a fault free storage space in each network shared by all the hosts within the same network as well as router.
- Every network has its own fault tolerance mechanism to detect and recover the faulty hosts as well as implement network services efficiently in presence of host or link failure within the network.
- MAS has been installed on each host of the network where Mobile Agents may be executed.
- MAS has also been installed at the routers but router is responsible just to receive and send the MA, not to execute them.

2.2 Grouping of Hosts

Our protocol is based on grouping of hosts within the same network based on kind of services offered by them. Hosts are grouped logically and one of the group member works as In-charge. One host is part of one group only.

One group appears as a single host to other hosts of distributed system.

To keep records of each group and all MAs running on each group of network, a group table and an agent table are maintained in shared storage. *GroupTable* contains the information about which host is part of which group and which is the in-charge host of the group, while agent table stores the information about which mobile agent is running on each group as well as on which host. A Failed-HostTable for each group is also maintained to keep the record of failed or inaccessible host. Hosts within each group communicate with each other through Group Communication Services designed to operate in a Local Area Network [6], [9].

2.3 Layered Architecture

HFTP consists of three layers. Different kinds of faults are detected and recovered at different layers. These three layers have been implemented as proxy servers. Server at lowest layer is *Personal Daemon Server (PDS)* monitors the MA and MAS, server at middle layer is *Local Daemon Server (LDS),* monitors the hosts and server at highest layer is *Global Daemon Server (GDS)*, responsible for fault free migration of MAs in global network. LDS and PDS have been installed on each host of the network. GDS is installed on routers.We briefly describe the functionality of each server (see Figure-1).

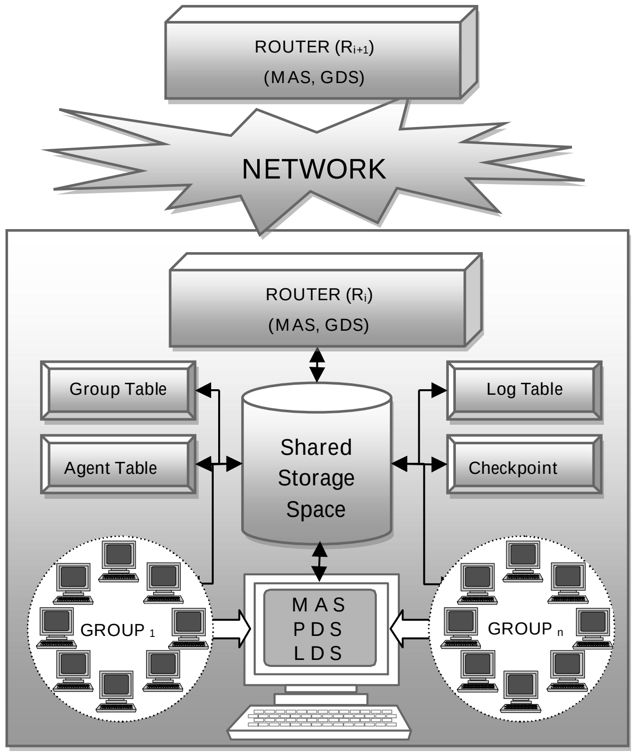

Fig. 1. Architecture of Hierarchical Fault Tolerance Protocol

Personal Daemon Server (PDS):

It monitors the MAS as well as all the MAs running on the host by maintaining a thread fro each. In case MAS crashes due to any reason, it is responsible to inform other group members about this fault as well as responsible to initiate recovery of MAS.

Local Daemon Server (LDS):

It is responsible for detecting the host failure as well as executing all group communication services within the group. LDS installed at Group in-Charge is responsible to assign a host to the MA submitted to the group impartially and balancing the load of each hosts. In case, either a host or MAS installed on host fails, it is responsible to distribute the load of failed host among active members of group.

Global Daemon Server (GDS):

It is responsible for receiving the Mobile Agents from other networks and then passing them to the appropriate group of its own network. It log an arrival entry for each incoming MA to the network into LogTable, also a departure entry is logged for each migrating MA. These entries in log table are used for recovery in case MA is lost during migration. It is also responsible to perform all function required for fault tolerant migration of MAs in the global network of networks by implementing rear guard concepts. In case all members of a group fail, it recovers MAs running in failed group from last checkpoint state.

3 Failure Cases And Recovery Procedure

In the following section we give the various faults that may occur during the life cycle of a MA and the schemes used by HFTP to tolerate them.

3.1 Mobile Agent Failure

MA is a piece of code which itself may fail during its execution due to some programming/data error or due to some uncaught exception. This fault can be minimized by providing the exception handling code for all known cases. But still if it fails, its fault is detected by the PDS which then rollback all uncommitted transactions and send a request to its MAS to create a new MA to carry the partial result to the user.

3.2 Mobile Agent System Failure

MAS or any of its components may fail/crash due to overload, resource unavailability, programming error or due to some other reasons as it is also a piece of code. Due to failure of MAS, all MAs running on it will get lost.

PDS watches MAS by maintaining a thread for it. MAS crash is recognize by PDS which then informs its LDS about this failure and reloads the MAS. LDS informs other members as well as in-charge of the group which then distributes the load of failed system among other active members. Newly selected hosts recover the MAs from last checkpoint state [2] and continue their execution.

3.3 Host Failure

Host is a machine in the network that provides a logical execution environment MAS to host and execute MAs. It may go down at any time; consequently MAS as well as all the MAs running on that system will get lost. All group members watch each other and fault of a host within the group is detected by other members of the group. After detecting the host failure, in-charge distributes the load of failed host among remaining group members. If failed host is the in-charge, then remaining members of the group will cooperatively elect a new group in-charge based on predefined priorities. If failed host is the only host in the group then GDS is responsible to perform the recovery. Failed host is recovered by using network own fault tolerance mechanism.

3.4 Communication Link Failure

All 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 mobile agent will be lost in its way. In order to tolerate link failure, HFTP uses two different strategies. For agent transfer within the network TCP has been used. TCP has been designed specially for a reliable, point-to-point, and sequenced communication; it suits agent transfer functionality very nicely. Link failure within the network is tolerated by TCP as it can recover an agent loss during migration from one host to another within LAN. Link failure may affect the group communication services to execute and make it difficult to detect the faults. We assume that group communication services efficiently implemented within the network and LAN is capable to tolerate such faults.

For agent transfer in the global network UDP has been used. Since MA may be lost during the migration from one router to other, precaution has been taken to avoid transferring MA to a failed host by providing alternative list of hosts to be visited by MA [11], [12], [13]. GDS installed at router is responsible to implement Migration protocol.

Migration protocol

Migration of mobile agent from one network (router) to other uses migration protocol, which is inspired by the concept of rear guard [4], [5], [15]. In this approach one witness agent is created to watch the actual agent. This agent ensures that MA successfully received at destination router (see Figure-2).

Assume that currently mobile agent is at the router R_{i-1} and ready to migrate to router R_i, but before migrating MA first spawn [5] a witness agent at R_{i-1} and then migrate to R_i. On arrival of MA at R_i its GDS first log an entry log^i_{arrive} in the log table, saves the agent and then send an acknowledgement message MSG^i_{ack} to R_{i-1}. Witness agent at Ri-1 is waiting for acknowledgement and if there is no fault, it will receive the message on time. The case that the witness agent at Ri-1 fails to receive acknowledgement includes –

a. Message lost due to unreliable network
b. Message arrive after time out
c. MA is lost due to unreliable network after leaving R_i and before arriving at R_{i+1}

All components at router is reliable except the link, so MA or Acknowledgement is lost due to link failure or delayed due to network traffic.

Once fault is suspected by the witness agent, it takes following actions-

1. Creates a probe (another agent), which travels to R_{i+1}. It carries checkpoint data with it as it may require recovering lost agent for case 1.
2. For the next two cases probe will find a log entry in the log table of R_{i+1} and just resend MSG_i^{ack} to R_i.
3. For case 1, probe will first recover lost agent and then send MSG_i^{ack} to R_i.
4. After sending the probe witness agent again
5. waits for acknowledgement.

If again it does not receive acknowledgement, it assumes the network fault and wait for network to resume

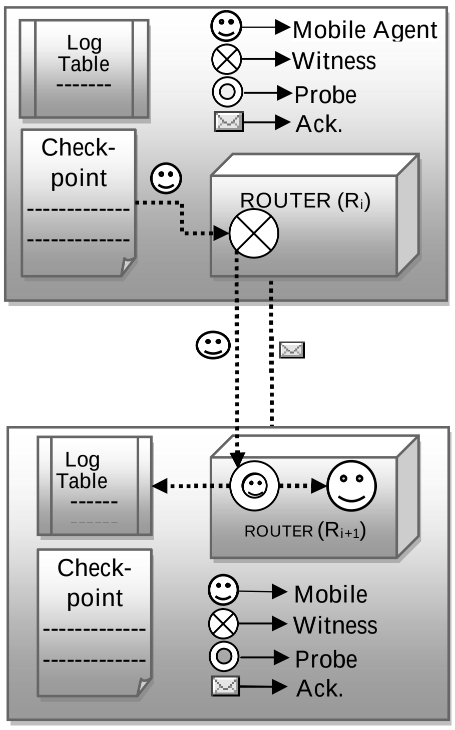

Fig. 2. Migration Protocol

and keeps on sending probe until receives the acknowledgement or detects fault of router.

4 Modelling with Colour Petri Net (CPN)

CPN [3], [7], [8], [14] is a very powerful, interactive and widely used tool for modeling and simulations. It supports hierarchical designed as well. Its support for markup language provides the flexibility to model different components of the system according to user's requirements. By using different features of the CPN tools various components of HFTP has been declared and protocols has been modeled by using many places and transitions organized as hierarchy of pages. Concepts of substitution pages and fusion places have also been used to model the hierarchical behavior of the protocol.

Various tools provided by CPN such as monitoring, state space and user controlled simulation tools have been used to check the correctness of the modeled system. Various data gathering and report generation tools have also been used to generate and collect the data required for analysis.

4.1 Performance Analysis

Before starting the simulation, some parameters are required to be assumed while some are generated randomly or calculated during simulation. The assignment is based on the assumption that packet transmission time is fixed and it is independent of place, time or load of network. The MA takes constant time to execute on any host.

Transmission time for MA	= 200 TU
Transmission time for Acknowledgement	= 100 TU
Logging (Arrival / Departure) time	= 50 TU
Host assignment for In-charge	= 50 TU
Execution Time for MA / host	= 450 TU
Recovery time for Mobile Agent	= 50 TU
Time to Checkpoint data and state	= 100 TU

4.2 Overhead of using HFTP

Every fault tolerance mechanism adds some overhead to the existing systems in terms of time, space or requirement to maintain reliability. In order to observe the overhead due to HFTP, in terms of MA trip time and network overhead generated by it, we have modeled a protocol having no support for fault tolerance (without HFTP) and then compared the performance of the system using HFTP in a fault-free environment.

Figure-3 shows that trip time increases linearly for both HFTP and without HFTP. Since simulation has been performed in an ideal fault-free environment, here all the steps including execution and migration of MAs takes constant time. Trip time is higher for HFTP because it requires logging the arrival and departure at the router, also the in-charge has to execute a deterministic algorithm to assign a host to the arrived MA in the group. Checkpointing is also required, even if no faults occur during MA execution.

Figure-4 shows that, although network overhead increases linearly for both cases, HFTP generates more overhead as the number of servers increase. This is because HFTP requires sending an acknowledgement for every migrating MA to detect link failure. Number of acknowledgements increases with the number of servers in the MA itinerary. Further implementation of group communication services generates network overhead in Local Area Networks.

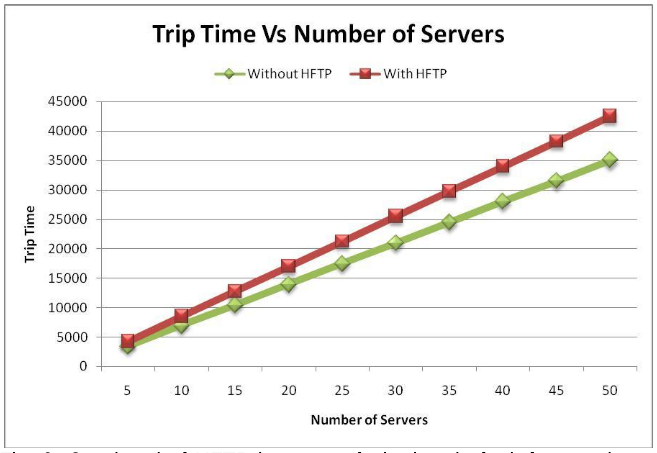

Fig. 3: Overhead of HFTP in terms of trip time in fault free environment

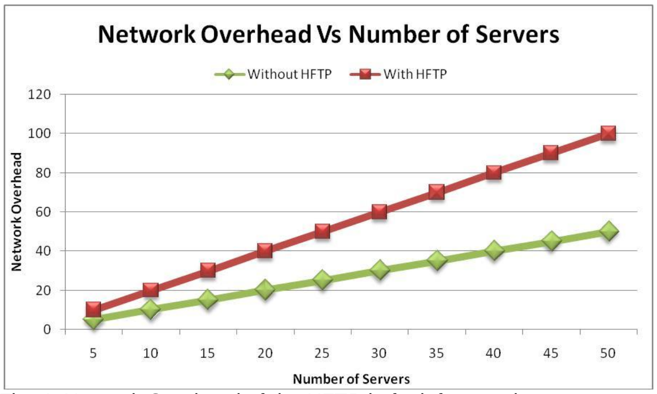

Fig. 4: Network Overhead of the HFTP in fault free environment

4.3 Fault cases and tolerance through HFTP

In order to observe the performance of HFTP in the presence of faults, we have generated various faults in the CPN model of HFTP by changing the failure probability rate and then measured its performance in terms of trip time, network overhead and number of execution steps.

In a real system, many faults may occur simultaneously, but for performance measurement we have generated one fault at a time. For each case, a MA with ten servers in its itinerary is launched. Simulation has been repeated hundred times and its average value has been used to predict the performance pattern.

Case 1: Mobile Agent System Failure

The MAS fails during execution of a MA according to its failure probability rate. Here it is assumed that in spite of failure, there is always at least one active host within each group to share the load of the failed host and MA does not get blocked.

Figure-5 shows that system failure rate is tolerated by HFTP. The global network overhead remains constant, while the local network overhead increases exponentially with failure rate, because a recovered agent may fail again and again. Every time a failed agent recovers, it adds some network overhead locally as it is required to transfer the recovered agent to its new host.

Figure- 6 show that trip time increases exponentially as failure rate increases, because every time the system fails, more time and extra execution steps required for detecting the fault, recovering the agent and resuming its execution on the new host. For small failure rates, the performance does not degrade too much.

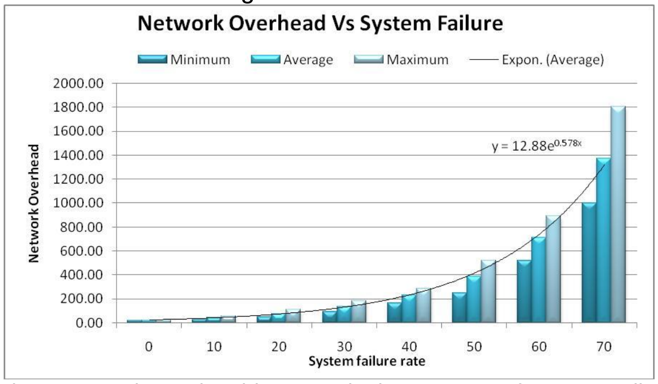

Fig. 5: Network Overhead by HFTP in the presence of System Failure

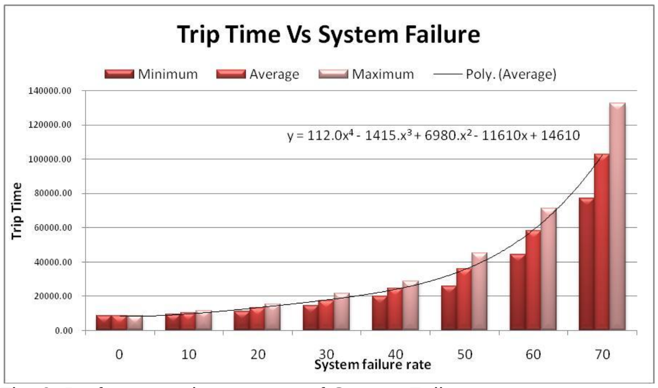

Fig. 6: Performance in presence of System Failure

Case 2: Host Failure

During the execution of MA, the host machine may go down and all MAs hosted by it are lost. HFTP tolerates host failure provided there is at least one active host per group to avoid blocking. Host failure is tolerated in the same way as system failure so the performance is expected to be similar as in case of system failure.

Unlike system failure, where a fault is detected by a thread, host failure is detected by other members of the group watching it. The fault detection mechanism does not increase any load, as it is always watching the hosts. Again, in case of host failure, only local network overhead increases, global package transfer remains constant.

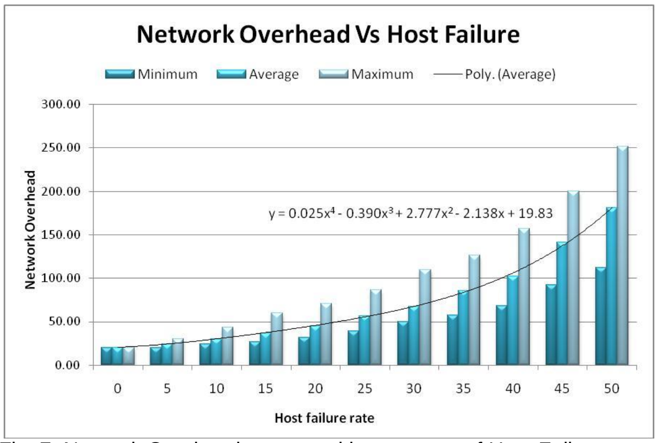

Fig. 7: Network Overhead generated in presence of Host Failure

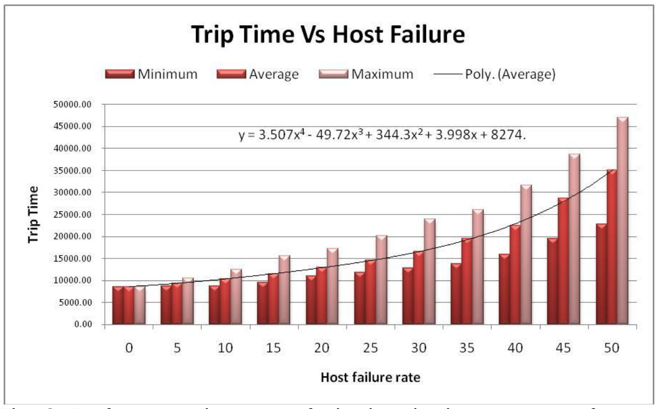

Fig. 8: Performance in terms of trip time in the presence of Host Failure

Figure-7 shows that local network overhead increases almost exponentially with host failure rate. Result is same as in case of system failure, but slightly better as recovery of a failed host is the responsibility of network manager and has not been considered while system recovery is initiated by PDS. Figures-8 verify our claim that host failure is tolerated by HFTP and gives a similar performance as in case of system failure provided blocking does not occur. Trip time grows polynomial with host failure rate.

Case 3: Link Failure

A link may fail during the migration of a MA from one host to another within a Local Area Network or between networks. Due to link failure, a MA may get lost on its way. HFTP tolerates link failure unless it leads to network partitioning. Link failure during migration within a network is tolerated by using TCP and has not been used for performance analysis. Due to link failure, a MA or acknowledgement may get lost in a global network and require retransmission of acknowledgements or probes, which not only increase network overhead and execution steps but also trip time. Since failure is detected only after waiting time is over, so delay increases more as compared to network overhead and number of execution steps.

Figure-9 shows the pattern of network growth overhead as link failure rate increases. It also proves our claim that

HFTP is able to tolerate link failure. However when failure rate is more than 25%, overhead increase significantly. Figures-10 shows that if failure rate is more than 25%, performance of the system degraded significantly and more time is required while for low failure rate performance is comparable.

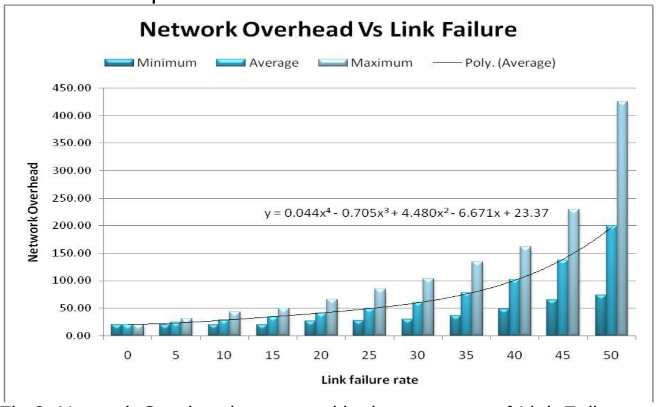

Fig.9: Network Overhead generated in the presence of Link Failure

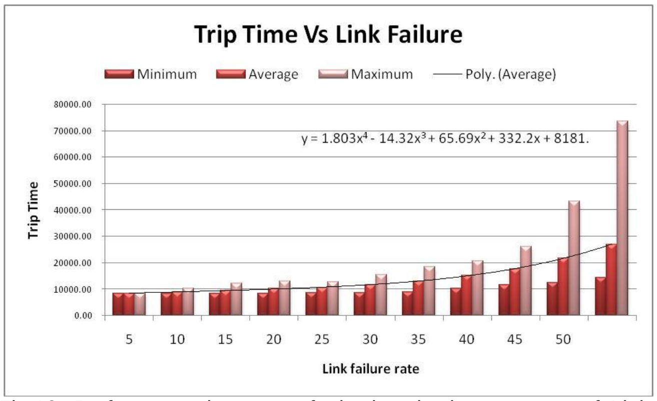

Fig.10: Performance in terms of trip time in the presence of Link Failure

Case 4: Agent Failure
Since MA failure detection and recovery takes place only at the host, it does not increase network overhead and not been observed. Figure-11 shows that trip increases almost linearly and performance is not degraded much until failure rate goes beyond 30%.

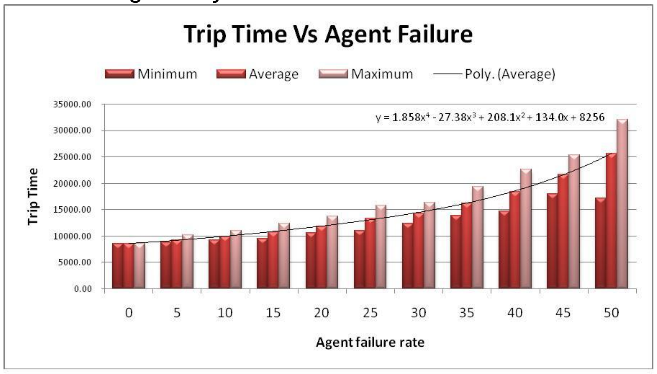

Fig. 11: Performance in terms of trip time in the presence of Agent Failure

5 Conclusion

The results show that HFTP is able to tolerate all kinds of faults without degrading the performance significantly. For low failure rate, the survivability of MA in HFTP is ensured and it is able to achieve tolerance without increasing network overhead or time delay substantially.
If host/system failure rate increases, then the MA may be blocked within a group. This blocking may be avoided by properly selecting the group size. But these failures are not frequent so the results are acceptable. Link failures in the global network may lead to network partitioning. This extreme case of link failure is tolerated by HFTP, if an alternative list of hosts is defined in its itinerary.
Also, if the order of the itinerary is not fixed, the MA can visit some other host in its itinerary and may try to visit the disconnected host latter when at least one of the links resumes. In the worst case when all the target hosts are disconnected with current network, MA will be blocked within the network.

References

1. M. Dalmeijer, E. Rietjens, M. Soede, D. K. Hammer, and A. Aerts, "A reliable mobile agent architecture", 1st IEEE International Symposium on Object-Oriented Real-time distributed Computing pp. 64-72, 1998.
2. 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.
3. K. Jensen, "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use", Volume 1,2 and 3, Monographs in Theoretical Computer Science, Springer-Verlag. ISBN:3-540-60943-1, 1997.
4. 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, Austin, Texas, USA, pp. 180-189, June 1999.
5. 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.
6. S. Mishra, Y. Huang, "Fault Tolerance in Agent-Based Computing Systems", Proceedings of the 13th ISCA International Conference on Parallel & Distributed Computing, Las Vegas, N V. August 2000.
7. Ouyang, C. and Billington, J.,"On verifying the Internet Open Trading Protocol", In Proceedings of EC-Web 2003, Lecture Notes in Computer Science 2738, Springer-Verlag, 2003, 292-302
8. Ouyang, C. and Billington, J., "An improved formal specification of the Internet Open Trading Protocol", In Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 2004, 779-783
9. S. Mishra, "Agent Fault Tolerance Using Group Communication", Proceedings of the 2001 International Conference on Parallel & Distributed processing Techniques and Application Las Vegas, N V. June 2001.
10. H. Pals, S. Petri, and C. Grewe, "FANTOMAS:Fault Tolerance for Mobile Agents in Clusters", Proceedings International Parallel and Distributed Processing Symposium (IPDPS), 2000, pp. 1236-1247, 2002.
11. 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.

12. R. B. Patel, K. Garg, " Fault-Tolerant Mobile Agents Computing On Open Networks", in Proceedings of Autonomic Applications workshop in conjunction with the International Conference on High Performance Computing (AAW-HiPC2003), Hyderabad, India. Dec.17-20, 2003, pp.296-308.
13. R.B. Patel "Design and Implementation of a Secure Mobile Agent Platform for Distributed Computing", Ph.D thesis, Department of Electronics and Computer Engineering and, Indian Institute of Technology, Roorkee, 2004.
14. H. Pathak, 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.
15. S. Petri, C. Grewe, "A Fault-Tolerant Approach for Mobile Agents", In Dependable Computing, Third European Dependable Computing Conference (EDCC-3) Prague, September 1999.
16. S. Pleisch, "Fault-Tolerant and Transactional Mobile Agent Execution", Ph.D thesis, Ecole Polytechnique Federal of Lausanne, Faculty of Information & Communications, 2002.
17. K. Rothermel, M. Straßer, "A protocol for providing the exactly-once property of mobile agents", In Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS), Purdue University, West Lafayette, Indiana, USA, pp.100-108, October 1998.