CHAPTER 6

FAILURE RECOVERY MODEL FOR FREE ROAMING MOBILE AGENTS

6.1 INTRODUCTION

A mobile agent is a distributed program which can move autonomously in a network, to perform its tasks, on behalf of the user. Mobile agents operate in an open environment to retrieve information that is distributed in the network. During migration, the mobile agents and their data are susceptible to failures, due to the faults in the network components, like the machine and the channel. It is challenging to address the mobile agent failure scenario that occurs while it migrates. This chapter presents a failure recovery model through information sharing for mobile agents that operate in a multi-region environment.

This model ensures the recovery of the lost mobile agent and the data by allowing a replication of the actual mobile agent at the previously visited nodes. However, unlike the existing methods, in this failure recovery process, the replication is temporary and limited to an optimal value. This is achieved by making the mobile agents that operate in a multi-region environment, to track, communicate and share the retrieved information between them. After the information sharing, the number of replicas to be maintained in each region is reduced to one. Furthermore, as there is at least one node in each region that holds the replica, this model guarantees that the information collected at each server is not lost. In addition, this model
survives region failure too, when implemented in a multi-region environment. Experiments were conducted in a distributed environment, and the results obtained are analyzed and compared with those of the existing methods, based on different testing parameters.

The following assumptions are made in this study, to clarify subsequent requirements that may arise, when addressing the failure recovery model presented in this section.

- **Agent server failure is temporary and recovered autonomously.**

  It is assumed that, the mobile agent has nothing to do with the recovery of a failed server; instead, the recovery of a mobile agent due to server failure is discussed. If the server failure is permanent, then the mobile agent is not able to retrieve the required information from it. In that case, the mobile agent is unsuccessful in achieving its task. There may be a situation in which the server failure is not permanent, and it recovers automatically, but the recovery time is not known. In this situation also, as the agent has no idea about the waiting time, the migration time of the mobile agent is not guaranteed. Thus, it is necessary for a node to recover autonomously after its failure. A mobile agent is allowed to revisit a node if it is not available during the mobile agent’s previous visit. However, the node is expected to recover before the mobile agent migrates back to its initiator.

- **Static itinerary planning is used for mobile agent migration.**

  For a mobile agent that uses a dynamic travel plan, as mentioned in Chapter 4, one of the node selection criteria is based on its alive state. That means, the mobile agent will migrate only to a node that is alive at that time. At the same time, for a mobile agent that uses a static travel plan, the alive state of a node may change at any time. A node, among the sequence of
nodes, can fail after the mobile agent starts its travel. Furthermore, in a static travel plan, it is not possible for a mobile agent to change the sequence or order of visit. In this context, it is justified to deal with the failure recovery in a static travel plan rather than in a dynamic travel plan.

- **All the location servers will not fail simultaneously.**

The failure of all location servers simultaneously, a possible but rare scenario, is not considered for the failure recovery scheme discussed in this section.

### 6.2 MULTI-REGION INFORMATION SHARING BASED FAILURE RECOVERY

The functional architecture of the Multi-Region Information Sharing based failure recovery using Location Server (MISM-LS) is shown in Figure 6.1. The information retrieval process is initiated by the agent service center. To begin with, it creates mobile agents based on the client’s request. The travel plan of the mobile agent is designed based on the nodes that are identified by the itinerary planner to visit. If the number of nodes to be visited is very large, multiple mobile agents are created in order to complete the task in quick time. From Figure 6.1, it is seen that, multiple regions are formed in order to accommodate these mobile agents in parallel and these regions are identified as Region$_1$, Region$_2$, ... , Region$_n$. A region (R) is established with a variable number of nodes, in order to distribute the mobile agents and thus restrict the roaming of the mobile agent to a limited number of nodes. Furthermore, the information sharing model introduced in this section can be effectively implemented in the multi-region environment.
In order to monitor and provide service to the mobile agents that are roaming in these regions, a centralized server referred to as the Location Server (LS), is assigned. The service of the LS includes, recording the location updates sent by the mobile agent, and sending the current location of a mobile agent that belongs to one group to another mobile agent of the same group. The LS forwards this location information to a mobile agent on request or voluntarily during the failure situation. The agent service center identifies the nodes to be visited by the mobile agent; in Figure 6.1 these nodes are represented as $S_{11}, S_{12},...,S_{16}$. The mobile agents that are dispatched
to the regions based on a task are grouped, and this group is given a unique identity. The individual agents of this group, which are intended to perform the information retrieval process, are dispatched to visit the identified nodes, one per region. The mobile agent visits these nodes one after another, as per the travel plan and performs its function in the nodes according to the task assigned to it. During the travel, after completing the task at a particular node and before migrating to the next node, each mobile agent updates its location to the LS.

This process guarantees that, at any given time, the current location of the mobile agents that are roaming in a region is available with the LS. In addition, using this location log, a mobile agent can locate and communicate with another member of its group and share the information retrieved by it up to that node. This information sharing process is done periodically, and the duration is fixed by the agent service centre. It is obvious that the information sharing process results in replication. At the same time, this type of replication plays an important role in achieving mobile agent failure recovery that is discussed in this chapter.

6.2.1 Information Sharing

The multi-region information sharing model with location server approach is a branch of the witness agent approach, and due to this the initial mobile agent (MA) migration and execution process begin in the same way as the witness agent approach. After being dispatched by the Location Server (LS) to the first node, the MA performs its task in the current server, logs a copy of it with data, updates its location with the LS, and migrates to the next server. The same process continues in all the nodes that the mobile agent visits. By this, after each migration, the data collected up to that level is available in the previous node. For example, (as shown in Figure 6.2) the entire data collected from $S_1$ and $S_2$ is available at $S_2$, and the data from $S_1$ to
Figure 6.2  Sequence of Mobile Agent Migration with Data and Information Sharing
S₃ at S₃ and so on. This is applicable to other regions also. However, unlike other existing models, the MISM-LS model encourages a mobile agent to share its information with other agents of its group during its travel. This sharing of information is done after visiting a threshold number of nodes in their regions, between a pair of mobile agents. For example, after visiting three nodes, a mobile agent of one region needs to share the data collected by it so far, with a MA of its group roaming at another region. Step 3 of Figure 6.2 shows how this information sharing takes place between the mobile agent residing in S₃ of region₁ and the mobile agent residing in N₃ of region₂. In this situation it should be noted, that after a mutual exchange, the data collected by the MA of region 1 and 2 is now available at both the nodes S₃ and N₃.

On the other hand, according to the multi-region information sharing approach, after the information sharing, the replications available in all the previous nodes are destroyed. That means, the replications logged by the mobile agent of region₁ (MA₁) at S₁ and S₂ of region₁ (and N₁, N₂ of region₂) are destroyed; however, these data are available at S₃ of R₁ and N₃ of R₂. Similarly, the data retrieved from N₁, N₂ and N₃ are available at N₃ of R₂ and S₃ of R₁. By continuing this process, after the next sharing, that is after visiting another three nodes, the data collected so far from both the R₁ and R₂ will be available at S₆ and N₆. If more regions are involved in this task, then after the information sharing, the overall data is available at the corresponding nodes of each region. This way of information sharing leads to a state where, the number of nodes in a region that holds the cumulative data, immediately after the information sharing, is one. At the same time, before the information sharing, the number of nodes that holds the replica depends on the number of nodes already visited in that region. Eventually, the total number of nodes that holds the cumulative data is ‘m’, if the number of regions involved in the information retrieval process is ‘m’.
6.2.2 Failure Detection and Recovery

The failure of a node during the execution of a mobile agent results in the loss of the agent as well as the data. In this failure recovery scenario too, the MISM-LS approach initially uses the witness agent approach. Consider a situation in which the mobile agent is available at $S_3$ after completing its task up to the node $S_3$. At this juncture, information sharing takes place as the mobile agent reaches the threshold value. As a result, the retrieved cumulative data of $R_1$ and $R_2$ is now available at $N_3$ and $S_3$. Now, the MA migrates to $S_4$ and during its execution at $S_4$, the node $S_4$ fails. Subsequently, the MA$_1$ and the cumulative data available at $S_4$ also fail and are lost forever. This is the stage at which the mobile agent failure recovery process should take place. In that way, it is essential to detect the failure before the implementation of a failure recovery procedure.

The failure of the mobile agent can be detected by different ways. First, there will be no location update by the failed mobile agent for a long time. After waiting for a threshold time period, the LS considers this long delay in the update as a mobile agent failure. Second, there will be no information exchange enquiry from the mobile agent of a region to other mobile agents of its group. This is identified by a MA that receives an outdated / repeated location from the location server for a location enquiry. Either it receives an address with which information sharing had taken place previously, or it receives no address from the LS. Third, within a region, this failure is identified by the previous server, as there will be no ping message to check the ‘aliveness’ of the corresponding server. The waiting time after which the mobile agent is considered as failed, is fixed by the agent service center. The agent service center fixes this by analyzing the average time taken by the mobile agent for migration and processing.
The MISM-LS model uses cloning methodology to recover the failed mobile agent. If the location server identifies the failure of a MA, it creates a clone of the corresponding mobile agent and dispatches it to the appropriate region. This is possible as the LS already has the actual mobile agent’s copy, that is dispatched to the nodes of that region. The LS dispatches the newly created copy of the mobile agent to a node from where it receives the last location request. From that node, the MA continues its travel as usual. This ensures the failure identification and recovery of the mobile agent.

Presume another situation; the node $S_5$ failed during the execution of the mobile agent. In this situation, the recovery is straight forward as the copy of the mobile agent available at $S_4$ will come into act. In continuation of the monitoring activity, the clone available at $S_4$ finds that it does not receive the periodic ping message from $S_5$. Immediately, the cloning recovery process takes place, and the newly created mobile agent resumes its function from $S_5$. The next scenario is one, in which both $S_5$ and $S_4$ fail. Still the mobile agent can be recovered from that of $S_3$ as stated previously. Now, the real critical situation arises, when $S_5$, $S_4$ and $S_3$ fail. It is not possible to perform the recovery using the MA available at $S_2$. The replica actually logged at $S_2$ has already been destroyed after the information sharing process between $S_3$ and $N_3$. This is the real scenario at which the model MISM-LS is benefited. Now the mobile agent is recovered using the replica available at $N_3$. After recovery, the mobile agent resumes its actual travel from $S_3$. This shows that, the failure of the mobile agent up to the information sharing process is recovered through the witness agent design, and after the information sharing, through the MA of the other regions. Thus, the MISM-LS approach withstands the single, continuous and complete failure of the nodes of a region. In fact, whenever the mobile agent is recovered, it is recovered with the data.
6.2.3 Benefits of the MISM-LS Model

The implementation of the failure recovery model, Multi-region Information Sharing Model using Location Server (MISM-LS) benefits the user in many ways when compared to the other existing models. The failure recovery models considered for the comparative study are, mobile time out design, mobile shadow design, and witness agent design. First of all, the mobile time out design does not support any replication scheme, and at the same time, does not withstand single node failure. Only one replica needs to be maintained for successful failure recovery in the mobile shadow design. However, this design does not withstand the failure of two continuous servers. In the witness agent design, the replicas to be maintained are ‘n-1’, (n - number of servers visited) which results in excessive replication (n-1 servers hold replication). Consequently, a certain amount of storage is consumed by the replicas at each node. In addition, this storage space increases with the itinerary, whenever the mobile agent retrieves information from the nodes, and hence, the total amount of storage consumed is very large in size. At the same time, the number of replicas to be maintained in MISM-LS model is quite limited, and it depends on the threshold value for information sharing. For example, from Figure 6.2, the required number of replicas for failure recovery is three, which is always fixed and constant. Thus, the MISM-LS model is beneficial in terms of both the number of replicas and the amount of storage usage.

Second, the network load due to the ping messages transmitted between the actual and the witness agents that check each other’s aliveness is considered. Here too, the issue of network load due to ping messages does not arise in mobile time out design. Similarly, in mobile shadow design the network load is much less, as this model experiences only a single ping message at any given time. However, both these models do not survive node
failures. At the same time, the witness agent design results in increasing network load. Here, it is essential for all the nodes visited by the MA, to transmit the ping message periodically, to its successor and predecessor; furthermore, this transmission of ping messages continues until the mobile agent successfully migrates back to its owner. In contrast, the MISM-LS model comfortably reduces this network load and in this model, the witness agent dependency is, once again, based on the threshold value for information sharing. Furthermore, this model consumes additional network load only at the time of information sharing between the agents of two regions. This network load does not make any impact as the information sharing is done at a comfortable time interval and between two regions at different timings. Based on these evaluations, it is clear that this model benefits the user in the context of the number of replicas, storage usage and network load. However, the success of the MISM-LS model comes at the cost of the following limitations:

- One important requirement for this model is synchronous information sharing. That means, information sharing is possible only after the agents of two regions reach the threshold value in their region. Even though one agent reaches its threshold value for information sharing it must wait for the other agent also to reach the threshold value. This requirement clearly affects the autonomous nature of the mobile agent.

- The MISM-LS model depends entirely on the reliability of the location server. In other words, the location server is subject to single point failure. The failure of the location server leads to the failure of the entire recovery process.
The size of the mobile agent as well as the replication increases in multiples after every information sharing. When the network load is reduced in terms of ping messages, it is increased in terms of the mobile agent size.

In order to overcome these vulnerabilities, a failure recovery model with a distributed location server, referred to as the Agent Monitor (AM) is developed.

6.3 MOBILE AGENT FAILURE RECOVERY USING AGENT MONITOR

The Mobile Agent Failure Recovery model using Agent Monitor (MAFRAM) is a branch of the Multi-region Information Sharing Model using Location Server (MISM-LS) approach, and thus, the initial MA migration and execution process begin in the same way as the MISM-LS approach. The functional diagram of the agent monitor based failure recovery model is shown in Figure 6.3. This section discusses only how this model differs in its functioning from the MISM-LS model. The agent service center dispatches the MA to the agent monitor, a dedicated server allotted to each region to monitor the movement of the mobile agent in its region. Mobile agents update their location with this agent monitor, whenever they are ready to migrate to the next node. However, in contrast to the failure recovery model discussed in section 6.2, the MAFRAM model encourages the mobile agent to send the retrieved data to the agent monitor. As a result, the mobile agent migrates to the next node only with the state and code but not with the data. Furthermore, this migration to next node and the transmission of the data to the agent monitor is done in parallel. This is applicable to all other regions involved in this task. Subsequently, the current location of the MA and the data collected up to that level is now available with the agent monitor.
On the other hand, all the agent monitors form a ring overlay structure and communicate among them, to share the data available with them. To perform this data sharing, a special mobile agent, referred to as the Token Agent (TA), is circulated among the agent monitors. The function of this TA is to collect the data available in one agent monitor and share it with the next agent monitor. For example, as shown in Figure 6.3, the token agent collects the data from agent monitor 1 and delivers it to agent monitor 2. Next, the token agent collects the cumulative data from agent monitor 2, and delivers it to agent monitor 3 and so on. After each cycle of the TA, it is definite that the data collected by the mobile agents of each region will be available in all the agent monitors. This information sharing process results

![Figure 6.3 Mobile Agent Failure Recovery Model – Distributed Agent Monitor](image-url)
in data replication, which plays an important role in achieving fault tolerance. At the same time, the number of replicas maintained by the MAFRAM model is dependent on the number of regions / agent monitors involved. Furthermore, these replicas are maintained at the agent monitors and not at the nodes.

### 6.3.1 Failure Detection and Recovery: Mobile Agent

The failure scenario discussed in section 6.2.2 is considered for this approach also. To begin with, as shown in Figure 6.3, the MA visits the first node $S_1$ of region $R_1$, performs its task, sends the retrieved data with the location update to $AM_1$, and migrates to $S_2$. The same process is repeated in $S_2$ and $S_3$. Now, the mobile agent migrates to $S_4$ and during its execution there, the node fails. The mobile agent residing in $S_4$ fails due to this, and the data collected from $S_4$ by that mobile agent is lost forever. It must be noted here that, only the data retrieved from $S_4$ is lost, and not as in the MISM-LS model, in which the data retrieved from $S_1$ to $S_4$ is lost.

It is the responsibility of $AM_1$ to detect the failure of $MA_1$ at any stage. In this regard, $AM_1$ detects the failure of $MA_1$, if there is no location update / data transfer by $MA_1$ for a specified time period. The time period is fixed, based on the average time taken by the mobile agent to communicate with the agent monitor for the location update. The actual time period is fixed as twice the average communication time, to allow some network delay. On arrival at each server, the timer is reset to its original value, and the mobile agent is expected to perform its task within the given time period, but once the timer goes off, this model considers it as a mobile agent failure.
There might be a situation, in which there will be no location update for a timeout period for some reasons other than the actual node / mobile agent failure. One reason might be that the MA spends more time at a remote server, in processing the database of a terabyte size. Likewise, the mobile agent may be kept in the waiting state for a long time, due to the server load. Network traffic is obviously another reason for the delay. In all these situations, according to the location server, the mobile agent has failed, but actually it has not.

As a response, the location server of this model retransmits the newly created mobile agent for failure recovery. This sort of recovery, results in mobile agent duplication, and in the worst case, it may lead to an endless loop also. As a result, there exist a number of such duplicate mobile agents, which may travel through the network, causing unnecessary network load. Furthermore, such suspended agents may be left unattended in the remote servers also.

To address the issue of the suspended mobile agents that travel in the network, the timer concept is modified on a small scale. If the mobile agent is still alive, and working at a server even after the timer expires, the mobile agent is expected to send a heart beat message to the Location Server (LS). After this, the timer is reset and the MA continues its work at that server. Now, after waiting for a timeout, the location server looks for the location update, or a heart beat message from the mobile agent. If both are not received the mobile agent is considered to have failed. Otherwise, the location server continues waiting accordingly. Instead of retransmitting a new mobile agent for failure recovery as soon as the timer expires, the mobile agent is encouraged to send a heart-beat message periodically to the LS. Now the failure identification is not based on the timeout value, but on the heart-beat message. At the same time, a time out is fixed for the heart-beat
message, and this time out is double the average communication time between a remote server and the LS.

Once the failure detection process is complete, the next step is the recovery process. To recover the failed MA, after detecting the failure, the agent monitor creates a clone of the available mobile agent. The newly created mobile agent is dispatched to the address, from where the last location update was made. From that remote server, the mobile agent continues its travel. This agent monitor based failure recovery model survives any number of single or continuous node failures, as the actual MA and the data are available at the agent monitor. The failure of one or more nodes does not affect the migration of the mobile agent as the backup is always available at the agent monitor. Thus, at any given time, the data collected up to that period and the actual mobile agent can be retrieved from the agent monitor.

6.3.2 Failure Detection and Recovery: Agent Monitor

The mobile agent failure recovery process of this agent monitor based model depends solely on the agent monitor. The distributed agent monitor is responsible for the detection and recovery of the failed mobile agent, but this agent monitor is also subject to a single point failure. The failure of an agent monitor results in the complete loss of data and the MA failure of that region. However, the failure of an agent monitor can be detected by other agent monitors in the group. The failure is identified when an agent monitor does not receive the token agent even after the expiry of its circulation time. By circulation time, this model indicates the time normally taken by the TA to visit all the agent monitors in the ring. Based on the assumptions made in this model regarding the recovery of a failed node, during the next rotation of the token agent the lost data can be recovered by the failed agent monitor.
6.3.3 Failure Detection and Recovery: Token Agent

The failure of the token agent is identified by an agent monitor, as it does not receive the TA for a long time. After waiting for a threshold time period, the current agent monitor considers this delay as the token agent failure. Now, the current agent monitor sends the negative acknowledgement to all the other agent monitors. In turn, the agent monitors that are in the alive state, inform their status to all the other agent monitors. The agent monitor, which holds the immediate lesser identification, takes the responsibility of the token agent recovery. This agent monitor creates a copy of the token agent, which is available with it, and forwards it to the agent monitor from where it receives the negative acknowledgement. From that agent monitor, the token agent continues its work. Thus, the failure of the TA is detected and recovered.

6.3.4 Benefits of the MAFRAM model

The limitations of the MISM-LS model are overcome by the MAFRAM model in many ways. With regard to the number of replicas, this model maintains them only at the agent monitors and not in any of the nodes as in location server based model. In the agent monitor based model the number of replicas maintained is depends on the number of the regions involved in the task. Thus, it drastically reduces the replicas when compared to the MISM-LS model, in which the number of replicas depends on the threshold value for information sharing in each region.

Likewise, in the MAFRAM model there is no issue of the ping messages at the node level. Here, it is not necessary for a witness agent to monitor the actual agent, as this approach completely eliminates the witness agent dependency overhead. This results in a high reduction of the network load. However, the ping messages between the agent monitors affect the
overall network load. The agent monitor based model greatly reduces the total storage usage at the node level also, as it does not encourage any storage at the node, but only at the agent monitor.

The trip time taken by the mobile agent for task completion is greatly reduced in the MAFRAM model, both in a fault-free and in a failure situation. The time of revisiting for every failure, increases the overall trip time in the mobile time out design. The main problem with the mobile shadow design is, it does not withstand continuous node failure even though the cumulative migration time is relatively less. In the witness agent scheme, this trip time increases with every continuous failure. Furthermore, excessive replication, storage usage and network load due to ping messages, are the other drawbacks of the witness agent approach.

Even though the location server based model discussed in section 6.2 overcomes some of the issues found in the other models, the location server based model also has its own limitations. The major drawback of this model is, the overhead in mobile agent tracking. For all information sharing processes, the corresponding mobile agent needs to be tracked by requesting the location server. The time taken by, (i) the mobile agent to request the location server; (ii) the location server to reply to the mobile agent; and (iii) the mobile agent to share the data with the others, heavily increase the overall trip time of the mobile agent. The model presented in this section comfortably addresses all these limitations. Especially, the synchronous migration requirement found in the MISM-LS model is removed in this model.

The mobile agent of each region can roam autonomously without waiting for any event to occur, thus maintaining its asynchronous nature. Similarly, the agent monitor of the MAFRAM model is not subject to single point failure. The failure of agent monitor can always be recovered, as these
agent monitors are distributed in all regions. More importantly, the increase in the size of the MA after every visit to the node is completely eliminated in this approach. The increase in the size of the mobile agent during information retrieval is default in all the other models discussed here. Meanwhile, in the approach presented in this section, after every information retrieval the data is transmitted to the agent monitor, and this completely reduces the size of the mobile agent. From all these discussions, it is evident that the MAFRAM model works better in all aspects like, trip time, network load, storage usage and the number of replicas. Moreover, this model withstands any number of node failures, thus ensuring the successful migration of the mobile agent back to its owner.

In spite of all these benefits, this model comes at the cost of an additional agent monitor. The overhead in the inclusion of the additional agent monitor is not included in the performance analysis, as the main focus of this work is in the achievement of failure recovery. Further, the work aims to implement the failure recovery with less trip time, network load and storage usage. When dealing with large networks, the overhead on the additional agent monitor is accepted and ignored.

6.4 ANALYTICAL MODEL FOR FAILURE RECOVERY

In order to estimate the reliability of the mobile agent-based information retrieval system discussed in this chapter, this section discusses the analytical model. This model focuses on the reliability aspect of mobile agent-based system on the component reliability of the processors and communication links during the agent’s travel. The performance of the mobile agent may be strongly affected by the failure of any hosts or the communication link between any two hosts that are used during the agent’s travel. As a result, it is important to construct a reliable mobile agent system that withstands these issues. In the analytical model presented here, the
approach defined by Hsieh and Hsieh (2003) that is used to analyze the reliability of distributed system, is adopted. Furthermore, to extend this approach for the mobile agent system, the analytical approach defined by Mahmoud and Daoud (2006) is also adopted. The system reliability is defined as the product of the reliability of the components involved in the task. In the context of the system presented in this thesis, one component is the processor on which a mobile agent is executed as operational during the period of the mobile agent execution and the other component is the communication link used when the execution is operational.

6.4.1 Mobile Agent Reliability Formulation

Let ‘p_i’ be a processor located at the processing node n_i, where i = 1,2,...,k (k-total number of nodes involved in the task). It is assumed that each processor represents a processing node, without any loss of generality. Similarly, let L_{ij} be the communication link connecting two processing nodes n_i and n_j. The communication route Path_{ij} is defined as the path between n_i and n_j. As different speeds are associated with different processing nodes, the time required to execute the request of a mobile agent may vary from a node to another. Let t_{ij} represents the execution time of agent a_j at processing node n_i. The assignment matrix ‘A’ indicates whether the task is performed at a node or not.

The component reliability of a processor ‘p_i’ itself, defined as the probability of failure-free operation of ‘p_i’ over a specific period of time, is an exponential function of the negative product of failure intensity and time (Musa 1989). As a result, the component reliability function ‘R_i’ of ‘p_i’ used in the analytical model discussed here is given as,

$$R_i(A) = e^{-\int_0^t \lambda_i(u)du}$$  \hspace{1cm} (6.1)
where ‘t’ is the processing time and ‘λd(t)’ is the failure-intensity function of ‘pi’. In most of the cases, the failure rate is constant in time. If the failure rate is constant λd(t) = λi, then the reliability function is rewritten as,

\[ R_i(A) = e^{-\lambda_i t} \]  

(6.2)

Consequently the reliability of ‘pi’ during the execution of task ‘T’ based on the agent assignment ‘A’, is given as:

\[ R_i(A) = e^{-\lambda_i t_i^{(Total)}} \]  

(6.3)

It must be noted here that, \( R_i(A) = 1 \) when \( t_i^{(Total)} = 0 \). (That is, no work is assigned to processor ‘pi’ and \( t_i^{(Total)} \), indicates the total execution time of the mobile agent during its travel). At the home server ‘\( \eta_h \)’ only serialization (S) and de-serialization (D) processes are performed. No other tasks are performed at the home server. Now the component reliability ‘\( R_h \)’ at home server of ‘\( p_h \)’ is given as,

\[ R_h = e^{-\lambda_h (t^{(D)}_{h} - t^{(S)}_{h})} \]  

(6.4)

In the same way, the component reliability of the communication link ‘Lij’ during the agent’s task is defined as,

\[ R_{ij} = e^{-\lambda_{ij} t^{(C)}_{ij}} \]  

(6.5)

where ‘\( \lambda_{ij} \)’ is the failure rate of ‘\( E_{ij} \)’, and ‘\( t^{(C)}_{ij} \)’ is the communication time required to transmit the agent from node ‘\( \eta_i \)’ to node ‘\( \eta_j \)’.

The reliability of the communication path ‘\( \text{Path}_{ij} \)’ is defined as the product of the reliabilities of all its communication links. If the communication path is defined as, \( s_i \to s_i^1 \to s_i^2 \to \ldots \to s_i^n \to s_j \), then its reliability ‘\( R_{ij} \)’ is given as,

\[ R_{ij} = e^{\sum_{k=i}^{n} \lambda_{ij}^{(k+1)}} \]  

(6.6)
Finally, if the mobile agent travel path is defined as, \( s_h \rightarrow s_I \rightarrow s_2 \ldots \rightarrow s_k \rightarrow s_h \), then the reliability of the mobile agent with an assignment matrix `A` is given as:

\[
R(A) = (\prod_{i=1}^{k-1} R_i(A), R_{i(i+1)}, R_{jr}, R_{ji}, R_{jk}, R_k(A))
\]  

(6.7)

Based on the definitions and discussion of the mobile agent’s reliability calculation, the constant failure rate is defined as the number of failures per unit time.

\[
\text{Failure rate} = \lambda = \frac{f}{n}
\]

(6.8)

where, \( f \) = the total failures during a given time interval and \( n \) = the number of items placed on test.

### 6.5 EXPERIMENTAL RESULTS AND DISCUSSION

In this section, the experiment for the comparison of the five mobile agent failure recovery models is explained and analyzed comprehensively. It is proved that the agent monitor based failure recovery model is more reliable and better than other models, more particularly in the case of failure recovery with less overhead.

#### 6.5.1 Experimental Setup

The network for the experiments is constructed using fifty nodes that are distributed geographically and connected through the Internet. The configuration details of these nodes that are evolved from the fifteen basic physical servers are shown in Table 6.1. In order to strengthen the scalability issue, virtual machines are created in these physical servers, using VMware virtualization software. The general architecture of the VMware virtualization based agent execution environment on which the Aglets tool operates is shown in Figure 6.4.
Table 6.1 Configurations of the Servers used in the Experiments

<table>
<thead>
<tr>
<th>Server ID (Location)</th>
<th>Virtual Machine ID</th>
<th>IP Address</th>
<th>Host Processor / RAM</th>
<th>Distance (KM) approx.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS (Chennai, India)</td>
<td></td>
<td>192.168.1.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1 (Chennai, India)</td>
<td>1</td>
<td>128.57.108.112</td>
<td>1.2 GHz / 2 GB</td>
<td>&lt;1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>203.102.18.212</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>203.102.18.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>203.102.18.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2 (Chennai, India)</td>
<td>5</td>
<td>10.0.128.22</td>
<td>1.2 GHz / 3 GB</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>192.168.47.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>192.168.208.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3 (Chennai, India)</td>
<td>8</td>
<td>113.112.196.245</td>
<td>2.2 GHz / 3 GB</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10.160.81.102</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10.160.81.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>10.160.81.88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4 (Kavarapettai, India)</td>
<td>12</td>
<td>192.168.1.8</td>
<td>1.2 GHz / 2 GB</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>198.154.10.122</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>198.154.10.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>198.154.10.103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5 (Kavarapettai, India)</td>
<td>16</td>
<td>113.112.196.238</td>
<td>1.2 GHz / 2 GB</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>10.160.81.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>10.160.81.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6 (Kavarapettai, India)</td>
<td>19</td>
<td>113.112.196.248</td>
<td>2.5 GHz / 3 GB</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10.160.81.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>10.160.81.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>10.160.81.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7 (Bangalore, India)</td>
<td>23</td>
<td>210.212.247.83</td>
<td>2.2 GHz / 3 GB</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>24.102.174.65</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>24.102.174.68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.1 (Continued)

<table>
<thead>
<tr>
<th>Server ID (Location)</th>
<th>Virtual Machine ID</th>
<th>IP Address</th>
<th>Host Processor / RAM</th>
<th>Distance (KM) approx.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S8 (Bangalore, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>173.83.205.234</td>
<td>2.5 GHz / 3 GB</td>
<td>372</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>42.162.18.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>42.162.18.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S9 (Coimbatore, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>192.168.1.1</td>
<td>1.5 GHz / 2 GB</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>203.102.18.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>203.102.18.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S10 (Coimbatore, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>61.216.75.132</td>
<td>1.2 GHz / 2 GB</td>
<td>418</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>192.112.10.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>192.112.10.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S11 (Tirunelveli, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>110.234.80.36</td>
<td>1.5 GHz / 2 GB</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>10.7.121.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>10.7.121.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S12 (Tirunelveli, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>110.234.112.10</td>
<td>1.5 GHz / 2 GB</td>
<td>615</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>10.7.121.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>10.7.121.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S13 (Trichy, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>180.151.210.12</td>
<td>2.2 GHz / 3 GB</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>214.67.10.107</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>214.67.10.112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S14 (Trichy, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>180.151.120.8</td>
<td>2.5 GHz / 3 GB</td>
<td>315</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>214.67.10.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>214.67.10.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S15 (Trichy, India)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>180.151.124.128</td>
<td>1.2 GHz / 3 GB</td>
<td>308</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>214.67.10.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>214.67.10.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>214.67.10.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 6.4 Architecture of the Virtualization used in the Experiments

At each physical server, virtual machines are created with Linux or Windows XP as the guest operating system. A screen shot with three virtual machines created on a server is shown in Figure 6.5. On the other hand, all the virtual machines are considered as separate nodes, based on their IP address. However, to justify the bandwidth and migration time issues, in the experiments, mobile agents are made not to visit two virtual machines of the same node continuously.

The IBM aglets is used as the agent server to accommodate the mobile agent. The failure recovery models are tested for an application, where a group of research team members store different types of documents (technical papers, reports, personal notes, etc.) in their computers. These members reside in different and distant locations, and may want to retrieve / provide related information that may be available with / required by their team members.
A search can be launched by any one of the members to retrieve the required document information and wire transfer it to a local directory. The search process is started by launching an application from the ASC; it enters a search string, for instance, a key word. To determine the related terms from a file, a simple algorithm is used for a given string to be searched. To perform the search, the ASC creates mobile agents based on the requirements. These mobile agents migrate to the predetermined nodes one by one as per the itinerary plan. In each of the nodes, the mobile agent makes a local search and the relevant files or documents are collected and appended to the documents already retrieved from the previously visited nodes.

6.5.2 Reliability Evaluation

Before analyzing the performance of the failure recovery models, it is necessary to justify the need for a fault tolerant system. In order to do this, initially, the failure probabilities of the nodes are tested by dispatching a
single hop mobile agent to all the nodes available in region 1 and 2. The task assigned to the mobile agent is to visit ten nodes in a region. The success of the mobile agent in returning to the home server is recorded for repeated tests and the results attained are shown in Figure 6.6.

**Figure 6.6 Successful Migration of the Mobile Agent**

From Figure 6.6, it is seen that the ratio of the mobile agent’s success is relatively poor. It means that the reliability of the system is comparatively less. Out of ten tests, only six times in region 1 and four times in region 2, the mobile agent’s travel is a success. Link failure, compromised server, crash failure are some of the reasons for this failure. These node failures always result in mobile agent failure and data loss. This scenario reveals the necessity of fault tolerance for mobile agents.

The migration, processing and cloning times of the mobile agent during its travel from a source to a destination are recorded. This process is
repeated six times and the average values of the results are determined and used for the actual evaluation. A sample result of the agent migration between the nodes \( S_2(6) \) and \( S_6(19) \) is given in Table 6.2. The results shown in Figure 6.7 are applied to the Equation (6.8) in section 6.4.2; \( n = 20, \ f = 10 \), the failure rate = 0.5. Using this result, the reliability of the system is calculated. The reliability, as per Equation (6.2) of section 6.4.1 is given by, 0.6905. After repeated tests, the failure rates of the nodes used for the experiments are determined. Using the failure rate of each node given in Table 6.2, the reliability of the system considered for the work presented in this chapter is also determined.

Table 6.2 Mobile Agent Migration details from Server \( S_2 \) to \( S_{12} \)

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Migration time (seconds)</th>
<th>Processing time (seconds)</th>
<th>Cloning time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>18.124</td>
<td>8.125</td>
<td>3.102</td>
</tr>
<tr>
<td>T2</td>
<td>14.658</td>
<td>8.763</td>
<td>3.210</td>
</tr>
<tr>
<td>T3</td>
<td>12.273</td>
<td>8.342</td>
<td>3.218</td>
</tr>
<tr>
<td>T4</td>
<td>17.833</td>
<td>7.989</td>
<td>3.184</td>
</tr>
<tr>
<td>T5</td>
<td>12.921</td>
<td>8.056</td>
<td>3.192</td>
</tr>
<tr>
<td>T6</td>
<td>18.296</td>
<td>8.355</td>
<td>3.213</td>
</tr>
<tr>
<td>Average</td>
<td>15.684</td>
<td>8.272</td>
<td>3.187</td>
</tr>
</tbody>
</table>

Likewise, the serialization and de-serialization time of the mobile agent at each node is determined. The average value attained after repeated tests is kept as constant in order to simplify the reliability calculation. The serialization time is 0.82 seconds and de-serialization time is 0.54 seconds.
The reliability of a node during the execution of an agent is determined using the data given in Table 4.3 of Chapter 4. This is applied on Equation (6.3). For instance, the reliability of node S1 is given by 0.6046. At the same time, for the host node, Equation (6.4) yields 0.9732. The reliability of node S1 is the probability that node S1 will work for at least 50 hours. The serialization time is 0.82 seconds and de-serialization time is 0.54 seconds. Similarly, the reliability of the link is calculated, and the detailed description of the various reliability calculations, with regard to the mobile agent system presented in this thesis, is given in Table 6.3.

6.5.3 Performance Analysis

In order to analyze the performance of the failure recovery models discussed in this chapter, the main parameter considered is the overall round trip time. This round trip time includes the agent’s migration time from one node to another, and the data transfer time. The migration time between node Si and the node S_{i+1}, is decomposed into travel time and processing time. Suppose the probability density function is given by l(t), the travel time probability density functions are given by r(t) and r_i(t) and the processing time probability density function is given by e(t). Now the migration time for moving the agent from Si to S_{i+1}, denoted by l_i, can be expressed as l_i = \sum_{j=1}^{v_i} l_j, where v_i indicates the number of nodes that are selected by the mobile agent to visit.

In the event of a performance analysis, this thesis ignores the recovery of the host during its failure. That means, the mobile agent does not wait for a host to recover after the occurrence of its failure.
<table>
<thead>
<tr>
<th>Description</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
<th>S11</th>
<th>S12</th>
<th>S13</th>
<th>S14</th>
<th>S15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Agent Failure Rate</td>
<td>0.02</td>
<td>0.04</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.08</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.08</td>
<td>0.06</td>
<td>0.05</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>Mobile Agent Reliability</td>
<td>0.37</td>
<td>0.14</td>
<td>0.61</td>
<td>0.37</td>
<td>0.22</td>
<td>0.06</td>
<td>0.61</td>
<td>1.00</td>
<td>0.61</td>
<td>1.00</td>
<td>0.08</td>
<td>0.05</td>
<td>0.08</td>
<td>0.37</td>
<td>0.08</td>
</tr>
<tr>
<td>Node Reliability</td>
<td>0.60</td>
<td>0.14</td>
<td>0.91</td>
<td>0.39</td>
<td>0.20</td>
<td>0.21</td>
<td>0.29</td>
<td>1.00</td>
<td>0.63</td>
<td>1.00</td>
<td>0.13</td>
<td>0.17</td>
<td>0.38</td>
<td>0.29</td>
<td>0.28</td>
</tr>
<tr>
<td>Link Reliability (From S1)</td>
<td>-</td>
<td>0.83</td>
<td>0.82</td>
<td>0.98</td>
<td>0.97</td>
<td>0.93</td>
<td>0.88</td>
<td>0.60</td>
<td>0.83</td>
<td>0.41</td>
<td>0.32</td>
<td>0.59</td>
<td>0.59</td>
<td>0.65</td>
<td>0.48</td>
</tr>
</tbody>
</table>
If the explicit distribution of the processing time and travel time is known, for the witness agent approach, the Laplace transform distribution (Dweik et al 2012) of the migration time, $F^*(s)$, is given by,

$$F^*(s) = \frac{(1-\rho)R^*(s)E^*(s)}{1-\rho R^* s} \{1-[\rho(1 - \rho)R^*(s)]^i\}$$ \hspace{1cm} (6.9)

Where, $\rho$ - probability of failure, $R^*(s)$ – distribution of round trip time, $E^*(s)$ – distribution of execution time, $d$ – average network connectivity, $i$ – remote server number. Similarly, the round trip time for the witness agent approach, $R^*(s)$, is given by,

$$R^*(s) = \frac{(1-\rho)R^*(s)}{1-\rho(1-\rho)R^* s} \{1-[\rho(1 - \rho)R^*(s)]^i\}$$ \hspace{1cm} (6.10)

For the shadow approach, the same equation can be used for the calculation of the migration time by considering the value of $d=1$, as the shadow approach supports a single replica at the agent server. This is the case with the agent monitor based failure recovery model also, as the single replica is maintained at location server.

In addition, these tests are conducted for both the fault-free and failure scenarios. The failure situation is implemented by introducing faults at nodes that are selected randomly. That is, a node is programmed to fail and restart autonomously, for every predetermined time period. The value of the time period is selected randomly by the node itself. In order to clearly understand the difference between these models at the micro level, initially, the experiment is conducted with a limited number of servers; here it is six. The threshold value for information sharing is fixed as 2 for the MISM-LS model.
The various failure recovery models considered for the evaluation and comparative study are: Mobile time out design – Method 1; Mobile shadow design – Method 2; Witness agent design – Method 3; MISM-LS; and MAFRAM.

The performance of these methods has been evaluated using the following measures. For the results obtained, the corresponding graphs have also been plotted.

- **Trip time**: The total time taken by the mobile agent after dispatch and up to the time it returns to the creator.
- **Network load**: Total amount of data transmitted over the link during the trip time.
- **Storage usage**: Total amount of memory occupied by the actual data and the replicas at the nodes, agent monitors or the location server.
- **Number of replicas**: Total number of replicas maintained by the models during the entire information retrieval process.

### 6.5.4 Trip Time

The trip time is calculated as the sum of the agent’s migration time between the nodes, the processing time (information retrieval time) at the nodes, and the additional time taken for the recovery during failures. The lower the trip time the better the task completion for the time constraint applications. The recorded overall trip time in a fault free environment is shown in Figure 6.7. This round trip time is calculated based on the Equations (6.9) and (6.10) given in the previous section. The results based on the trip time demonstrate that the trip time of the agent monitor based model is lower than the methods 2, 3 and 4. However, the trip time of method 1 and
MAFRAM is same. It is observed that the difference in the trip time between the methods 2, and 3 and that of 1 and MAFRAM, is very low. This is because only the additional cloning time is included in the methods 2 and 3; further, this difference increases with the migration.

At the same time, MISM-LS shows a greater increase in the trip time, which is due to the additional time taken for the mobile agent tracking for information sharing. In addition, this increase goes higher after every information sharing. In spite of the equal performances the agent monitor based model includes an additional data transfer time. However, this time is ignored, as the migration of the mobile agent to the next node and the data to the location server, is done in parallel. The real difference between the failure recovery models can be determined only in the failure situations. The trip time of the mobile agent during the node failures at $S_3$ and $S_6$ is shown in Figure 6.8.

![Figure 6.7 Trip Time Comparison in the Fault-free Situation](image-url)
From Figure 6.8, it is seen that the increase in the trip time difference for method 1 during failure at S₃ and S₆, is very high. That is, the failure of server 3 causes method 1 to revisit the servers S₁ and S₂ only. Instead, the failure of S₆ causes it to revisit all the servers from S₁ to S₆, and this is the reason for the greater increase in the trip time. At the same time, if the failure occurs after visiting a large number of servers, this difference will become worse (see Figure 6.11). That situation makes method 1 to revisit all the servers, which definitely increases the trip time to a large extent.

In a single node failure scenario, the difference between the methods 2 and 4 is not much, as both these methods follow the shadow design. That is, when method 3 shows results similar to those of method 2 during fault-free situations (Figure 6.7), they show different results in failure

Figure 6.8  Trip time Comparison in Failure Situation (Failures of Servers S₃, S₆)
situations (Figure 6.8). During failures, method 3 takes more time, as it needs to migrate back to the previous nodes until it reaches an ‘alive’ node. Finally, the real effect of the MAFRAM method in a failure situation rather than in a fault-free situation is justified. Though there is not much difference between methods 2 and 4 in a single node failure situation, these methods differ considerably in continuous failure situations as shown in Figure 6.9.

![Figure 6.9 Trip Time Comparison in a Continuous Failure Situation (Failures of Servers S₅, S₄ and S₃)](image)

Methods 1 and 2 do not support continuous failure recovery, and are not shown in Figure 6.9. However, the actual benefit of MISM-LS over the methods 2 and 3 is clearly indicated in Figure 6.9. The claim here is, in a single server failure, methods 2 and 3 behave equally, whereas when multiple continuous failures occur, they differ. Similarly, the trip time for the location server based recovery is high over method 2 in a fault-free situation, but not much in single server failure, and much less in a continuous failure situation. (Figures 6.7, 6.8, 6.9). This is one significant factor to claim that MISM-LS performs better in a failure situation, rather than in a fault-free situation.
On the other hand, the MAFRAM method shows the lowest trip time in all situations (Figures 6.7, 6.8, 6.9). The cloning time, agent tracking time and information sharing time are not included in this model. The only additional time included in this model is the recovery time during failures. This is shown in Figure 6.8, between the nodes S_2, S_3 and S_5, S_6 and in Figure 6.9, between the nodes S_4 and S_5. On the whole, the total trip time taken by method 5 in a failure situation is much lesser than any of the other existing models.

To validate the effect of the trip time for a bigger architecture, the models are tested with repeated configurations of fifty servers. The resultant trip time of the mobile agent when there is no failure, is given in Figure 6.10, and with single failures at servers, S_5, S_{16}, S_{33} and S_{49} in Figure 6.11. Comparing Figure 6.8 and Figure 6.10, it is clear that in both cases, location server based recovery shows a higher trip time. That is, in a fault-free situation, MISM-LS takes the highest trip time in all the models. Likewise, method 1, MAFRAM and methods 2, and 3 also

![Figure 6.10 Trip Time Comparison in the Fault-free situation for Large Number of Servers](image-url)
behave equally for a large number of servers. However, comparing Figures 6.8 and 6.11, it is seen that MISM-LS behaves better than methods 1, and 3 for a lesser number of servers. In contrast, it behaves negatively for a large number of servers. In both the cases, the trip time is very high for method 1. More importantly, the MAFRAM model shows better results, when compared to the other models in all situations.

![Graph comparing trip times for different methods](image)

**Figure 6.11 Trip Time Comparison in a Failure Situation for Large Number of Servers (Failures of Servers $S_5$, $S_{16}$, $S_{33}$ and $S_{49}$)**

### 6.5.5 Network Load

In modeling the network load ($N$), the following assumptions have been made.

- Size of the Mobile Agent’s Code: $S_{AC}$
- Size of the Mobile Agent’s State: $S_{AS}$
- Size of the Mobile Agent’s Data: $S_{AD}$
- On a node the mobile agent’s data size is increased by $S_D$
- Cost of one ping message : $C_{PM}$
- Threshold value for Information sharing : $Q$
- Cost of message passing : $C_{MP}$

The equation used for the calculation of the network load is,

Migration from the home server to the first node

$$N_1 = S_{AC} + S_{AS} \text{ (Mobile agent is not taking any data from the home server to the first server)}$$  \hspace{1cm} (6.10)

Migration from the one node to another node \hspace{0.5cm} (X = S_{AD} + S_{D})

$$N_2 = S_{AC} + X + \sum_{k=0}^{n} \binom{n}{k} X^k a^n k S_{AS}, \text{ where, } k \text{ – node number, } n \text{ – total number of nodes, } a \text{ – remote server.}$$ \hspace{1cm} (6.11)

Migration from the last node to the home server

$$N_3 = S_{AD} + S_{AS} \text{ (Mobile agent’s code is not necessary to be migrated to the home server from the last node).}$$  \hspace{1cm} (6.12)

For the calculation of the network load, $N_1 + N_3$ is included for all the models. Apart from this, the network load calculation includes the following for each of the models,

Network load (N) calculation for,

Method 1 $\rightarrow$ Sum of $i=1$ to $n$ ($N_2$)  \hspace{1cm} (6.13)

Method 2 $\rightarrow$ Sum of $i=1$ to $n$ ($N_2 + C_{PM}$)  \hspace{1cm} (6.14)
Method 3 $\rightarrow$ Sum of $i=1$ to $n$ \((N2 + [(i-1) \times C_{PM}])\) \hfill (6.15)

\[\text{MISM-LS} \rightarrow \text{Int}(\frac{n}{Q}) \times \left[\text{Sum of } i=1 \text{ to } Q \ (N2 + [(i-1) \times C_{PM}]) \right] + 2 \times (C_{MP} + S_{AD}) \times \left[\text{Sum of } i=1 \text{ to } Q \ (N2 + [(i-1) \times C_{PM}]) \right] + 2 \times (C_{MP} + S_{AD}) + \text{abs}(\frac{n}{Q}) \times \left[\text{Sum of } i=1 \text{ to } Q \ (N2 + [(i-1) \times C_{PM}]) \right] \hfill (6.16)\]

\[\text{MAFRAM} \rightarrow \]

\[N1 = S_{AC} + S_{AS} \hfill (6.17)\]

\[N2 = S_{AC} + S_{D} + \sum_{k=0}^{n} \binom{n}{k} x^k d^{n-k} S_{AS}, \hfill (6.18)\]

\[N3 = S_{D} + S_{AS} \hfill (6.19)\]

In evaluating the failure recovery model’s performance based on the network load measure, the following considerations were made.

- The mobile agent’s initial code size is 500 Kilobytes; state size is 200 Kilobytes.
- The cost of message passing is 300 Kilobytes.
- At each node it is assumed that the size of the data retrieved by the mobile agent is 2 Megabytes.
- The cost of a ping message is 200 Kilobytes.
- The threshold value for information sharing is 2.

The result of the network load that is calculated based, on the definitions and equations given in this section for a fault-free environment, is shown in Figure 6.12.

From Figure 6.12 it is evident that, in a fault-free situation, the difference between the models is very low and in some cases almost equal.
The network load for method 1 depends solely on the increase in the mobile agent size. Method 2 varies very slightly from method 1, as it includes the load of a single ping message in addition. However, method 3 shows a little higher load, due to the necessary ping messages between all the nodes involved in the task. The main thing is, the network load increases with the itinerary.

**Figure 6.12 Network Load Comparison in a Fault-free Situation**

For MISM-LS, the network load increases with the itinerary just as in method 3, but suddenly drops to the lowest level after every information sharing. Considering MAFRAM, the network load is the same as in method 1. Though the data and state, and the code travel in parallel, the total network load consumed is equal to the size of a mobile agent with data. As a result, in a fault-free situation the real impact of MISM-LS is not clear except from method 3. The network load for a fault-free environment is compared with the experiment conducted with a large number of nodes. The attained results are shown in Figure. 6.13.
Figure 6.13  Cumulative Network Load for the Mobile Agent Migration in Fault-free environment

From Figure 6.13, it is clear that there is no difference in the results obtained for both the smaller and bigger architectural situations. As the difference between methods 2 and 3 is based on the number of ping messages, the variation is very small. Similarly, if the average network load is considered, methods 3 and 4 show the same results. However, the real difference between these two models is in the information sharing. At the same time, agent monitor based model shows much less network load compared to all other models. In this model, the retrieved data is not accumulated with the mobile agent; instead, it is delivered to the agent monitor as soon as it is retrieved. Conversely, in all other models the data is accumulated with the mobile agent. Due to this, for each migration the network load increases for all the other models, but remains constant in MAFRAM. (Note that, the results discussed here are only based on the assumed data. Fluctuations and differences may occur when the actual data
size is taken into account. However, it is obvious that the results of the actual data size affect all the results equally).

In order to analyze the behavior of these models in a failure situation, tests were conducted in a failure environment also. The network load comparison by considering the failures of $S_3$ and $S_6$ is shown in Figure 6.14.

![Cumulative Network Load for the Mobile Agent Migration in a Failure environment](image)

**Figure 6.14** Cumulative Network Load for the Mobile Agent Migration in a Failure environment

For method 1, the network load increase is steady, as it contains only the increase in the size of the agent. At servers $S_3$ and $S_6$, the increase is high, due to the revisit of all the servers. Method 2 shows a considerable decrease compared to method 1, for it includes only one ping message at any given time. The continuous transmission of the ping messages between all $n-1$ servers until the success of the mobile agent, makes the network load of method 3, the highest among all. Considering MISM-LS, it starts equally with all the others, increases gradually in a fault-free situation, and steadily in
a failure situation. The increase is due to the information exchange scenario; otherwise, it remains constant. This indicates that the network load for MISM-LS is better, when compared to method 3. The MAFRAM approach shows the lowest network load, as this approach eliminates the ping message, information sharing process, and MA migration with data, completely.

Finally, the network load of MAFRAM is much less, as this model performs the mobile agent migration and data dispatch in parallel. The mobile agent size does not at all increase for every migration, as it immediately forwards the collected data to the LS. Furthermore, during failure recovery also, the LS is directly involved in the recovery process, and every failure is independently addressed. In addition, the shadow or cloning concept is completely eliminated. These advantages of the MAFRAM make it work better in respect of network load also.

6.5.6 Storage Usage

Storage usage is defined as the amount of store consumed by an mobile agent to log its data at each node for failure recovery. The total storage usage between the mobile agent’s start and the arrival at the home server or location server, is shown in Figure 6.15. This parameter is not applicable for method 1, as the data or the mobile agent is not logged in any nodes. For method 2, at any given time, the replica of the mobile agent at that state is available in the current and previous nodes. The total store used by this model is 282 megabytes at the time of completion. This approach withstands single node failure, but not multiple continuous failures. Even though method 3 withstands multiple continuous failures, the storage usage is the highest among all other models. The storage usage of MISM-LS reaches a higher level before the information sharing process, and reaches the method 2 level after the information sharing process.
Figure 6.15  Comparison of the Cumulative Storage Space Consumed for Failure Recovery

This indicates that MISM-LS works better, if the failure occurs immediately after the information sharing process. For the MAFRAM model, the data is not stored in any node but only at the agent monitor. The sum of the data available at the agent monitors after each agent migration, is the total storage used by this method. From Figure 6.15, it is seen that the graph for MISM-LS and MISM-LS stops at node S8, because these two methods are implemented in a multi region environment. By default, the number of nodes visited is not 8 but 15 at that level. In storage usage also the MAFRAM method performs better in most of the situations. As the MA moves only with the code, and this code is also to be executed in the execution environment, MAFRAM is relatively free from the storage usage overhead.
6.5.7 Number of Replicas

The total number of replicas to be maintained by each method for failure recovery is shown in Figure 6.16. For method 1, this parameter is not applicable. For method 2, it is one at a time and for method 3, it is ‘n-1’ for ‘n’ servers visited. For MISM-LS, it varies, based on the threshold value for information sharing. For the MAFRAM method, the replica is one for each region, but for the agent monitors it depends on its quantity. For this work, two regions of six servers were used, and the threshold value for information sharing is defined as two.

![Figure 6.16 Number of Replicas Maintained by Servers for Mobile Agent Failure Recovery](image-url)
6.6 SUMMARY

- The recovery of a mobile agent with data, which fails due to node failure is discussed, using two failure recovery models.

- The survival of the mobile agent in various failure scenarios such as single node failure, multiple node failures, and multiple continuous node failures is explained.

- The failure recovery of the various components of the failure recovery model, like agent failure, agent monitor failure, and token agent failure is also demonstrated with possible failure situations.

- The failure recovery model is tested in a distributed environment, that consists of a number of nodes with various configurations for the parameters, trip time, network load, storage usage and the number of replicas.

- For all the parameters, the agent monitor based failure recovery model performs better by recovering the failed mobile agent in less time. The network load caused due to this recovery is also reduced, when compared to the other models. Similarly, the cost of storage usage and the number of replicas are less in this approach.

- To strengthen the scalability issues, the MAFRAM model is tested with a large number of nodes. To implement this model, VMware virtualization software is used by deriving fifty nodes from fifteen physical servers.

- It was proved that the results attained with a smaller number of nodes clearly reflect in a large number of nodes also. This
ensures that the performance of the MAFRAM model is consistent for any number of nodes.

- Whenever the mobile agent is recovered from failure it was ensured that the mobile agent is recovered with the data.

- The implementation of the model presented in this chapter comes with the cost of the addition of the corresponding agent monitors.