Fault Tolerant Mobile Agent Systems by using Witness Agents and Probes

Sareh Beheshti *, Mansoureh Ghiasabadi **, Mona Sharifnejad * and Ali Movaghar ***

* Islamic ARAK AZAD University
  s-behesht@iau-arak.ac.ir
  m-sharifneiad@iau-arak.ac.ir

** Islamic ARAK AZAD University
  m-ghias@iau-arak.ac.ir

***Sharif University of Technology
  movaghar@sharif.edu

Abstract: In Mobile agents systems, agents can perform their programs with traveling from the server to others and fault tolerance is important in their itinerary. In the paper, existent methods of fault tolerance in mobile agents are considering and described. Then the method is considered that which uses cooperating two agents to fault tolerance, to detect and to recover server and agent failure, meaning three type of agents involved: actual agent which performs programs for its owner, witness agent which monitors the actual agent and the witness agent after itself, and probe which is sent for recovery the actual agent or the witness agent on the side of the witness agent. Communication mechanism is message passing between these agents. Traveling agent through servers, the witness is created by actual agent. Scenarios of failure and recovery of server and agent are discussed in the method. During performing the actual agent, the witness agents and probe are increased by addition the servers. Proposed scheme is that minimizes the witness agents as far as possible. Number of created witnesses and number of created probes until any server are considered and evaluated too. This method is Simulated by C-Sim.

Key words: fault tolerance, mobile agents, probe, witness.

INTRODUCTION

Mobile agents are software programs that travel through servers for to perform their programs on behalf of own user and fault tolerance is fundamental and important in their itinerary.

In the through their live cycle may happen failures [14]. The failures in a mobile agent system, may lead to a partial or complete loss of the agent. So we should create the fault tolerant mobile agent systems. Failures could detected and recovered. Many of Methods exist relative to fault tolerance in mobile agent systems that is considered any of them.

Then the method is described and evaluated that which uses cooperating two agents to fault tolerance, to detect server and agent failure and to recovery mobile agent and witness agent, meaning three type of agents involved: the actual agent which performs programs for its owner, the witness agent which monitors the actual agent and the witness agent after itself, and probe which is sent for recovery the actual agent or the witness agent on the side of the witness agent [19]. A message passing mechanism exist between the actual agent and the witness agents to detect the failures of agent and recovery. In the method, a new witness agent is created by actual agent on current server when the actual agent want travel to the next server, in order to the witness monitors the actual agent. The witness knows arriving to a server and leaving it by the messages.

During performing the actual agent, the witness agents are increased by addition the servers, meaning any witness monitors the next witness and the actual
agent. Proposed scheme is that addition minimization of witness agents as far as possible, number of created witnesses and number of created probes until any server will consider and evaluate, because by comparision, could concluded that existing all of witness agent is not necessary on the initial servers. Simulation of this method is done by C-Sim [9].

The rest of the paper is strutured as follows. Section 2 presents the method of fault tolerance in mobile agent systems by using the witness agents and probes. In Section 3, we describe the Proposed scheme and its evaluation for minimization of the witness. Section 4 consider and evaluate number of created witnesses and number of created probes until any server. Finally, section 5 is identified conclusion and future work.

1. Fault tolerant mobile agent systems by using witness agents and probes

1.1. Relative methods

Many methods exit for fault tolerance in mobile agents, for example primary-backup models, Exactly-once model [20], fault tolerant mobile agent system based on the Agent-Dependent approach [13], mobile agent fault tolerance by exception handling approach [11], Fault Tolerance in Mobile Agent Systems by Cooperating the Witness Agents [19] and etc. The last method will describe in following.

1.2. System architecture

In this design, in order to detect the failures of an actual agent as well as recover the failed actual agent, another types of agents are designated, namely the witness agent and probe, that witness agent can monitor whether the actual agent is alive or dead, and probe can recover witness agent and actual agent that witness agent send it to relative server. A communication mechanism is between both types of agents. Every server has to log the actions performed by an agent. The information logged by the agent is vital for failure detection as well as recovery. The stable storage is need too. The overall design of the server architecture is shown in Figure 1.

If the actual agent is at server \( S_i \), the witness agent is at server \( S_{i-1} \). The actual agent and the witness agent are labeled as \( \propto \) and \( W_{i-1} \), respectively.

The behavior of the actual agent \( \propto \) is discussed first. After \( \propto \) has arrived at \( S_i \), it immediately logs a message, \( Log_{\text{arrive}} \), on the permanent storage in \( S_i \). The purpose of this message is to let the coming witness agent know that \( \propto \) has successfully landed on this server. Next, \( \propto \) informs \( W_{i-1} \) that is has arrived at \( S_i \) safely by sending a message, \( msg_{\text{arrive}} \), to \( S_{i-1} \).

\( \propto \) performs the computations delegated by the owner on \( S_i \). When it finishes, it immediately checkpoints its internal data to the permanent storage of \( S_i \). Then, it logs a message \( Log_{\text{leave}} \) in \( S_i \). The purpose of this message is to let the coming witness agent know that \( \propto \) has completed its computation and it is ready to travel to the next server \( S_{i+1} \). Then, \( \propto \) sends \( W_{i-1} \) a message, \( msg_{\text{leave}} \), in order to inform \( W_{i-1} \) that \( \propto \) is ready to leave \( S_i \). At last, \( \propto \) leaves \( S_i \) and travels to \( S_{i+1} \). Failure and recovery scenarios exit that the witness agent receive \( msg_{\text{arrive}} \) and \( msg_{\text{leave}} \) that they use the probe for recovery them.

1.3. creartion of the witness agent

The witness agent at server \( S_i \), \( W_i \), is spawned by the actual agent \( \propto \) after it logs \( Log_{\text{leave}} \) and before it moves to the next server \( S_{i+1} \). The youngest (i.e., the most recently) witness agent is monitoring the actual agent. On the other hand, the older witness agents are responsible for monitoring the witness agent that is just one server closer to the actual agent in its itinerary. Chain of the witness agent is

\[
W_0 \rightarrow W_1 \rightarrow W_2 \rightarrow \ldots \rightarrow W_{i-1} \rightarrow W_i \rightarrow \propto ,
\]

Where "\( \rightarrow \)" represents the monitoring relation. The above dependency is witnessing dependency. This procedure consumes a lot of resources along the itinerary of the actual agent. Then, we must decrease the number of alive witnesses as far as possible and minimize the witnessing dependency.

2. Proposed scheme

For the minimization of the witnessing dependency, if we assume that no \( k \) or more servers can fail at the same period of time, dependency is simplify by the witness length less than or equal to \( k \).
If the actual agent $\propto$ is at server $S_i$, the minimized dependency becomes:

$$\begin{align*}
&\text{if}(i \leq k) \quad W_0 \rightarrow W_1 \rightarrow \ldots \rightarrow W_i \rightarrow \propto \\
&\text{else} \quad W_{r-k} \rightarrow W_{r-k+1} \rightarrow \ldots \rightarrow W_i \rightarrow \propto
\end{align*}$$

Then, $k$ witness agents are sufficient to guarantee the availability of the actual agent. The number of servers is variable in various mobile agent systems and $k$ is variable too ($k$ meaning the number of required witness agents). As the result, we must obtain number minimum of the alive witness agent (i.e., $k$ witness agent).

### 2.1. Description of algorithm

In this paper, is assumed that number of servers is 50. This method is simulated by C-Sim.

This algorithm, is assumed that its topology is linear and the mobile agent starts from the first server until it reaches final server. Method of execution is considered in the three level namely level$_0$, level$_1$ and level$_2$.

In the level$_0$, execution of mobile agent is without fault tolerance and the agents fail because of server failure, meaning if the agent is on the server that it is failed, execution of mobile agent become passive at the result.

level$_1$, server failure is detected and recovered too. Recovery of failure server is done after the specific time. In this level, if the server is failed and the mobile agent reside on that server, it become passive and then become active after recovery of server and after the time namely repair.

In the level$_2$, when the server is failed, the server and the mobile agent are recovered. In the level is used the witness agent and probe for recovery of agent. If server goes to down state and any mobile agent, witness or probe are located at it, then they lost and disable. The mobile agent checks the status of next server and if it is down, waits until repaired server wakes it up.

### 2.2. Comparing and evaluation of reliability charts

As compared with level$_0$, reliability of level$_1$ decreases by increment of the servers, because while the agent (in server $S_j$) is waiting for recovery of server $S_{j+1}$, there is probability that a failure happens to the server where the agent resides, but in level$_0$ is without fault tolerance, as the result of there is not repair of server (figure 2).

In level$_2$, charts of reliability for the mobile agent are shown in figure 3 and 4 (for 50 server) which they are obtained with number of alive witness agents number $k=1$ to $k=10$. As compared of charts, we can result that number of witness agent that are required is 6, because reliability mobile agent in all of the servers is 1, means number minimum of witness agent is 6.
2.3. Evaluation of created witnesses based on charts

In this part, charts of created witnesses until any server are describe by using number of witnesses from 1 to 10 and are evaluated, that figure 5 is shown them.

![Figure 5- chart of created witnesses with until any server k=1,2,3,4,5,6,7,8,9,10](chart5.png)

In the chart with k=1, the number of created witnesses until any server increases by increment of the number of servers and it is incremental and linear, Because number of created witness agent to any server equal number of that server by increment of the number of servers.

The charts with k=2 to k=10 are incremental and no linear , meaning the number of created witnesses until any server increases by increment of the number of servers.

2.4. Evaluation of created probes based on charts

In this part, charts of created probes until any server are describe by using number of witness from 1 to 10 and are evaluated.

![Figure 6- chart of created probes with until any server k=1,2,3,4,5,6,7,8,9,10](chart6.png)

Figure 6 is shown charts of created probes until any server. All of them are incremental and no linear wit k=1 to k=10, meaning number of created probes until any server increases by increment of the number of witness agents and number of servers.

3. Conclusion & Future work

In the last part, execution of mobile agent is described in the three level and reliability of it and related of charts are presented.

As compared with level\textsubscript{0}, reliability of level\textsubscript{1} decreases by increment of the servers, because while the agent (in server \(S_i\)) is waiting for recovery of server \(S_{i+1}\), there is probability that a failure happens to the server where the agent resides, but in level\textsubscript{0} is whitout fault tolerance, as the result, there is not any repair of required server. In level\textsubscript{2}, we have wanted to obtain number minimum of alive witness agent along the itinerary of the actual agent for 50 servers and we have found that reliability of mobile agent become 1 with 6 alive witness agent.

Generally, number of created witness agents until any server increases by increment of the number of servers, but if number of active witness agent is 1, then the relative chart is linear. Because number of created witness agent to any server equal number of that server by increment of the number of servers.

Number of created probes until any server increases by increment of the number of witness agents and number of servers.

In the considered method, its topology is linear, number of mobile agent equal 1 and traveling of mobile agent is sequential through servers. In the Future work, because of simulation executions become closer to reality, we can consider following assumptions:

- non-linear topology : Topology of this algorithm is linear and the mobile agent travels through servers consequentiality. we can change this topology from linear to non-linear (for example mesh).

- Increment of mobile agents : we can increase the number of mobile agents. The mobile agents can devide to different sets that any of them have different parameters of functional.

- about witness agent: 1) first mobile agent that reach to any server, create a witness agent on the server. The witness monitor all of the mobile agents which reach to the relative server. 2) in the beginning, a witness agent is created by any server and on the any server and it monitors any mobile agent that travels to the relative server (if the mobile agents are consistent).3) if the mobile agents are not consistent , a witness agent monitors any set of the consistent mobile agent that travels to the relative server.

REFERENCES


