International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 4, Number 4 (2014), pp. 343-350

http://www. irphouse.com /ijict.htm

Comparative Analysis of Various Fault Tolerance Approaches in Mobile Agents System

Richa Mahajan[1], Gurpreet Singh[2], Ramandeep Kaur[3] and Rahul Hans[4]

[1, 2, 3]Dept of Computer Science and Engineering, G.N.D.U,
[1, 2, 3] Amritsar, India.
[4]Dept of Information and Technology, PTU,
[4] Jalandhar, India.

Abstract

Mobile Agent technology is emerging as a new paradigm in the area of distributed and mobile computing. Its paradigm has attracted many attentions but it is still not widely used. The reason for this is that it suffers from the various issues regarding reliable mechanisms like security and the fault tolerance in mobile agent system. Mobile agents have uniqueness that they can migrate from one server to another in order to satisfy requests made by their clients. Since mobile agent moves from one server to another in an itinerary it is easily prone to various faults like server crash or agent crash, which makes fault tolerance one of the main issues of reliability in mobile agent system. This paper surveys various fault tolerance techniques in mobile agent system proposed by various authors. These techniques are evaluated on the basis of defined parameters like type of fault, exactly once execution, agent centric, system centric, coordination and communication.

Keywords: Mobile Agent; Fault Tolerance; checkpoint; System Centric; Agent Centric;

1. Introduction

Mobile agents are software agents having a unique ability to migrate from one host to another in its itinerary. The main characteristic of mobile agent which differentiates it from other paradigms is

- Proxy: Mobile Agents may act on the behalf of someone.
- Reactive: Ability to sense environment and act accordingly.
- Autonomous: It means have an ability to act without direct external interfaces.
- Cooperative and Coordination: Mobile Agents should coordinate and co-operative with other agents to perform a particular task.
- Migrate: It is the core property of mobile agent that it can migrate or transport itself.

Mobile Agents can execute on those system which provides resources to it that are needed to complete its task. To accomplish their task mobile agent moves to remote host and can compute locally and only results can transfer through network, which results less congestion in network.

In mobile agent computing environment any component of the network machine, link, or agent may fail at any time, thus may preventing mobile agents from continuing their executions. Therefore, fault-tolerance is a vital issue for the deployment of mobile agent systems. Fault tolerance specifies an ability of a system to respond gracefully to an unexpected failure. Its aim is to provide reliable execution of agents even in case of failure. Two desire properties to achieve fault tolerance are Non-Blocking and Exactly Once.

While travelling within a network or from one network to another to complete its task there could be a possibility of failure. Various types of failure can occur in mobile agent system are discussed below [7]

- Node Failure: The Complete failure of a compute node implies the failure of all agent places and agents located on it.
- Agent Failure: Mobile agents can become faulty due to faulty computation, or other faults like node or network.
- Communication Failure: Failure of entire communication link or single link.
- Fault of component of the agent system: Failure of agent place or incomplete agent directory.
- Loss of message: this arises due to network failure or failure of communication unit of an agent.

The rest of the paper is organized as follow. Section 2 describes various existing fault tolerance techniques of mobile agent proposed by various authors. Like above section 3 discuss various fault tolerance techniques and evaluate them on the basis of defined parameters like type of fault, exactly once, agent centric, system centric and coordination and communication. Section 4 briefly discusses conclusions and future work.

2. Various Fault Tolerance Approaches

Most of the fault tolerance techniques in mobile agent system are based on replication and checkpointing. Replication based approaches are classified under two categories are Temporal Replication and Spatial Replication.

2.1 Using the CAMA Framework

In [4] authors discussed the CAMA framework supports application-level fault tolerance by providing a set of abstractions and a supporting middleware that allow developers to design elective error detection and recovery mechanisms.CAMA supports system fault tolerance through exception handling. There are three basic operations available to the CAMA agents for catching and raising exceptions are: raise, check and wait. These functionalities are complementary and orthogonal to the application level mechanism used for programming internal agent behaviour.

The advantage of this approach is that the exception handling allows fast and effective application recovery by supporting flexible choice of the handling scope and of the exception propagation policy and the drawback of this approach is that it can block the execution in case when an exception is raised to the agent which has left the scope.

2.2 Optimistic Replication Approach for transactional Mobile Agent Fault Tolerance

In [1] authors proposed an approach called optimistic temporal replication allowing multiple executions of the mobile agent to avoid blocking situation also considers transactional execution and semantic failures. It prevents a partial or complete loss of mobile agent [5][9]. Authors consider a transactional mobile agent system having components like Place (P) which consists of a lookup directory (LD) and storage unit (SU), Transaction Manager (TM), Mobile Agent (ma), Watch Agent (wa) and itinerary. This protocol in based on the behaviour of Mobile Agent (ma), Watch Agent (wa) and itinerary.

- Mobile Agent: It moves to the TM for register the transaction, spawns a new Watch Agent (wa_0) and returns to the first node to start its execution.
- Watch Agent: It doesn't participate to the transaction computation. It only listens to messages sent by the mobile agent.
- Transaction Manager (TM): It monitors the global distributed transaction execution.

This approach is based on check pointing, chain control and message passing to detect and recover failed agent. Multiple executions are detected using lookup directories and are solved at the destination place (TM) by committing only one execution. The advantage of this approach is that it doesn't violate the exactly once execution property by using a commit at destination protocol and the drawback of this approach is that it does not assume perfect failure detection and tolerate network partition.

2.3 Region-based Stage Construction Protocol for Fault tolerant Execution of Mobile Agent

In [6] replication-based fault tolerant protocols are classified into two approaches spatial replication based approach (SRBA) and Temporal replication based approach (TRBA). In SRBA the agent is replicated and sent to several sites so that the agent can

survive site failures. The Temporal approach is based on the check pointing the code and state of agent on the previous site [1].

SRBA has a drawback that additional communication cost is added when move to next stage. RBSC protocol is used for fault tolerant execution of mobile agents in a multi-region mobile agent computing environment. It uses new concepts of quasi-participant and sub stage in order to put together some places located in different regions within a stage in the same region. On a sequence of nodes a mobile agent ai executes its tasks. Each action that ai execute on a place p_i is called a step. Each step consists of a set of places called a stage Si [8]. Pw_i at S_i is called a worker, the others are called participants. When a worker fails, one of participants is elected as a new worker and takes over the action of the previous worker. In a multi-region mobile agent computing environment, places within a stage can be located in the same or different regions [8].

The advantage of this protocol is that it reduces the total execution time and decreases the overhead of stage works and the disadvantage is that an overhead occurs for stage constructed in the same region.

2.5 FATOMAS (Fault Tolerant Mobile Agent System)

In [3] authors introduce FATOMAS, a java-based fault–tolerant mobile agent system. There are two fault tolerant approaches i.e. Place dependent and Agent dependent.

FATOMAS is based on Agent dependent approach. This approach has the important advantage to allow fault – tolerant agent execution without having to modify the underlying mobile agent platform. Currently, FATOMAS supports Voyager mobile agent platform.

For enabling fault tolerance each mobile agent (called user agent), created a logger agent which is responsible for providing checkpointing and logging. A user agent and its logger agent form an agent pair. Logger agent doesn't participate actively in computing and needs only a small fraction of the available CPU capacity. User and logger agent monitor each other, and if a fault is detected by one of them, it can rebuild the other one from its local information.

The advantage is that logger agent uses only a small fraction of the available CPU capacity while providing fault tolerance to user agent and the disadvantage is overhead introduced by the replication mechanisms and with increasing the number of stages and the size of the agent.

2.6 Using the Witness Agents in 2-Dimensional Mesh Network

In [5] authors introduces that the server and agent failures are detected and recovered by the cooperation of agents with each other. In order to detect and recover the failed agent, another types of agent are used, namely the witness agent, to monitor whether the actual agent is alive or dead [9]. It prevents a partial or complete loss of mobile agent [1].

Three types of agents are:

- Actual agent: Agent which perform programs for its owner.
- Witness agent: Agent which monitors the actual agent and witness agent after itself.
- Probe: Agent which is sent for the recovery of actual agent or the witness agent.

A communication between both types of agents is done by sending Direct and Indirect messages [9]. When actual agent is unable to send a direct message to a witness agent for this purpose there is a mailbox at each server that keeps those unattended messages. These types of messages are called the Indirect Messages.

The advantage of this approach is that by the use of 2-D mesh network, dependencies among witness agent get reduced as compare to linear network and the drawback is that the existing procedure consumes a lot of resources along the itinerary of the actual agent as the itinerary becomes longer, more witness agents and probes are necessary, so system complexity increases.

2.7 Transient Fault Tolerance

In [2] author describes how to detect and recover random transient bit-errors at an agent before starting its execution at a host after its arrival at a host. mobile agent code often experience transient faults resulting in a partial or complete loss during execution at a host machine [2], [1], [5]. Errors are detected by comparing three images of code (original code and two replicas of it) and then recover them by applying XOR computation on them.

If one byte among the three bytes is corrupted, then this algorithm can detect and recover it. In case, damage of all the three images of an agent, HARD_ERROR is invoked for restarting or reloading the agent code execution from stable memory.

The advantage is it helps in detecting corrections and soft errors. fault tolerance is applied at low level (at every byte) which increases performance and the disadvantage of this error technique is it can't detect errors that can occur after the execution of an agent has started.

3. Comparative Analysis of Fault Tolerance Approaches

In this analysis and evaluation of the techniques is based on some parameters like type of fault, coordination and communication, agent centric, system centric, exactly once execution is shown in table 1. The parameters are discussed below:

- **Type of Fault:** It depicts which type of failure among agent failure, node failure, and communication failure.
- **Coordination & Communication:** It tells about the mode of coordination and communication i.e. Direct or Indirect.
- **Agent Centric:** Those approaches achieve fault tolerance by the mean of agent centric, marked by Yes else No.

- **System Centric:** Approaches that achieve fault tolerance with the help of mobile agents are system centric.
- **Exactly Once Execution:** It says that every transaction should be done only once.

Table 1: Comparative Analysis of Various Fault Tolerance Approaches Based on Following Parameters

Approaches / Parameters	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Type of fault	Agent	Agent & Link	Agent	Agent & Server	Agent & Server	Agent
Coordination & Communication	DR	DR	DR	DR	DR & INDR	DR
Agent Centric	Yes	Yes	Yes	Yes	No	Yes
System Centric	No	No	No	No	Yes	No
Exactly Once Execution	Yes	No	No	No	No	No

Table 2: Pros and Cons of various fault tolerance Approaches.

Mechanisms	**Pros**	**Cons**
Optimistic Replication Mechanism [1]	Exactly once doesn't violate and also helps to avoid blocking and network partitioning.	It deals with only semantic failures.
Transient Fault Tolerance [2]	Errors can be detected and corrected at bit level and provide countable performance.	Can't detect errors that may occur after an agent has started.
FATOMAS [3]	LA (Logger Agent) uses a little amount of CPU capacity while execution.	Overhead introduced by the replication and with increasing the number of stages and size of the agent.
CAMA Framework [4]	It provide framework to developers with a set of abstraction and handles fault tolerance by exception handling.	It just deals with fault tolerance at application level.
Using the Witness Agents [5]	Fault tolerance is achieved by cooperation of agents with each other.	Consumes a lot of resources along the itinerary.
Region-Based Stage Construction [6]	Reduce the overhead of executing regions located at different locations as a result execution time decreases.	At each time worker fails, quasi-participant replaces it's position with real- participant

4. Conclusion and Future Work

In this paper we have discussed various fault tolerance approaches proposed by various authors. All these approaches have their own pros, cons. Most of them suffer from a common problem that they violate exactly once execution.

From the future point of view work should be done to provide fault tolerance in dynamic applications and also avoid the violation of exactly once execution. For providing exactly once execution, at the time when a new agent arrives at the host the agent compares its composite key with the composite key of already executed agents. If on comparison both the keys are not same then the new agent will execute on the host and after successful execution it saves the composite key of recently executed agent at that host. If on comparison both the keys are same, it simply skips the computation at that host and jumps to next one. In this way one can achieve exactly once execution.

References

[1] Z. Linda and B. Nadjib, "Optimistic Replication Approach for Transactional Mobile Agent Fault Tolerance," In Proc. of 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Publication IEEE Conference, 2010.

[2] K.S. Goutam, "Transient Fault Tolerance in Mobile Agent Based Computing," In Proc. of INFOCOMP Journal of Computer Science, Vol. 4, page(s) 1-11, September 2005.

[3] K. Mohammadi and H. Hamidi, “Modeling of Fault-Tolerant Mobile Agents Execution in Distributed Systems,” In Proc. of Systems Communications, Publication IEEE Conference, 2005.

[4] A. Budi, I. Alexei and R. Alexander, “On using the CAMA framework for developing open mobile fault tolerant agent systems,” In Proc. of international workshop on Software engineering for large-scale multi-agent systems, Publication IEEE Conference, 2006

[5] A. Rostami, H. Rashidi and M. S. Zahraie, “Fault Tolerance Mobile Agent System Using Witness Agent in 2-Dimensional Mesh Network,” In Proc. of International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010

[6] S. J. Choi, M. S. Baik, H. S. Kim, J. W. Yoon, J. G. Shon and C. S. Hwang, "Region-based Stage Construction Protocol for Fault tolerant Execution of Mobile Agent,” In Proc. of the 18th International Conference on Advanced Information Networking and Application, Publication IEEE Conference, 2004.

[7] Yousuf, F. ; Zaman, Z. , “A Survey of Fault Tolerance Techniques in Mobile Agents and Mobile Agent Systems,” In Proc. of International Conference on Environmental and Computer Science, Page(s): 454 – 458, 2009.

[8] S. Pleisch and A. Schiper, “Modeling fault-tolerant mobile agent execution as a sequence of agreement problems,” In Proc. of 19th IEEE Symposium of RDS, Publication IEEE Conference , 2000.

[9] S. Beheshti and A. Movaghar, “Fault tolerance in Mobile Agent Systems by Cooperating the Witness Agents,” In Proc. of International Conference on Information and Communication Technologies, Vol. 2, Page(s): 3018 – 2, Publication IEEE Conference, 2006.