Review paper of novel clone fault tolerance in mobile agent

Manpreet Kaur[1], Ravneet Kaur Sidhu[2], Harkamal preet kaur[3]
Department of Computer Science Engineering
CTITR, Maqsaudan[1,2] , CTIEMT, Jalandhar[3]
[1] sahnipreet15@yahoo.com

Abstract: *Mobile agent is a wise operators in its agenda. Operators (agent) is an autonomous system which follows up for the benefit of the client(user). In the distributed computer environment, it has a wide extension and server can relocate starting with one host then onto the next host. In mobile computing environment, there are two primary issues happens; one is security and second is fault tolerance. Fault tolerance intends to give the solid execution of operators even in the event of disappointment or failure. To accomplish the fault tolerance, we actualize a novel clone fault tolerance strategy. Novel clone fault tolerance method utilizes the concept of check pointing, cloning the first operators (agent) and exactly once which implies the agenda is gone by at one time. The results are focused around the parameters like check pointing, round trip time and exactly once parameter.*

Keywords: *Mobile Agents, Check pointing, Cloning, fault tolerance and exactly once.*

1 Introduction:

Mobile agent innovation has a developing area of mobile computing environment. A Mobile agents are programs of software which follow up for the client to finish its assignment in its agenda. Mobile agent moves starting with one host then onto the next host in its agenda and executes just those framework in which to give assets to it that is expected to finish its assignment. An operator (agent) doesn't restart execution from the earliest starting point at new computer (machine); it proceeds where it exited off. Agents can be ordered along the few essential traits that the agent could show, for example, reactivity, pro reactivity, mobility and social-capacity. As for mobility, the agent has capacity to move around some computer (machine) system and relocation. There are two sorts of agents. Initially is mobile and other is stationary. The principle attributes of mobile agent which separates it from different standards are given below:

- **Proxy:** Mobile agents may act as the behalf of someone.
- **Reactive:** The Ability to sense environment and act accordingly.
- **Autonomous:** It means an ability to act without direct external interfaces.
- **Migrate:** It can migrate itself from one host to another host in its agenda.

Mobile agents have been created as an expansion to mobile code and its standard is not the same as others on the grounds that information as well as the code following up on the information is additionally transported among the hubs which makes the created application more adaptable. Mobile agents should replace the conventional client-server model and its architectures. In a client/server show, a server is a machine which gives some administration and a client makes demands for those administrations through a transmission channel communication between the client and the server is ordinarily through message passing. Accordingly, when a client

needs a specific administration, it generally sends a request message to the server that contains the required administration and may expand the systems activity. The mobile agent advances can be partitioned into two groups. In Weak mobility is just the code is moving, however no execution state is moving. In Strong mobility, code and also execution state is likewise moving with the agent program.

Mobile agent can relocate self-sufficiently among the nodes in a heterogeneous environment and can cooperate with other mobile agents. A mobile agent contains the accompanying 3 parts.

- **Code:** The program that can define the agent's behavior.
- **State:** The agent internal variables etc., which enable it to resume its activities after moving to another host.
- **Data:** Information describing the agent, its origin and owner, its history, resource requirements etc. Part of this may be accessible to the agent itself, but the agent must not be able to modify the attributes.

In distributing computing environment, there may be probability of connection, failures and communication happen in the execution of framework. The fault tolerance is an essential issue in a mobile agent. fault tolerance to react a framework nimbly in the event of failure. fault tolerance is important to convey the mobile agent framework. Two vital properties to attain to fault tolerance are:

Non-Blocking- It is utilized essentially for overcoming blocking issues posture by the exactly-once protocols. It is accomplished by either restarting a fails process or utilization copy agents that have the capacity assume control in time of essential agent failure without voting.

- **Exactly Once-** Fault tolerance prevents the system from partial loss of information by keep continues the execution of the machine.

Fault tolerance embrace the dynamically environment in its agenda. Fault tolerance procedure is focused around replication and check pointing and hybrid. Section 2 depicts diagram of a few procedures of Fault tolerance. Section 3 gives the results and evaluation. Section 4 provides the conclusion.

2 Fault Tolerance and existing approach

Fault tolerance empowers the framework to perform the continuously working even if there should arise an occurrence of intruded on things happened in the framework, it can never obstruct the framework. Fault tolerance is focused around two method i.e replication and check pointing.

Replication is a sort of repetition. A Replicated server are utilized to endure the issue. At the point when the server is down, utilization an alternate server which give the same administrations and proceed with the processing. Thusly, processing is not blocked when the server failure happen. It is of two sorts spatial and temporal.

Check pointing is a methodology of saving status data. It stores the gathered information at the home server after the specific interval. It additionally serves to diminish the execution time of an agent. However check pointing expands the aggregate time of execution if the issue is not happened then it makes the overheads.

In novel clone fault tolerance method, we utilizes the concept of checkpoint and exactly once. The first agent is relocate starting with one host then onto the next in its agenda however in the event that the first

agent is neglected to execute then the clone of that unique agent will executed on that server and proceed with their execution. In novel clone check pointing is utilized to store the information after some server. It lesser the round trip time of the agent to execute.

It begins the execution where the server is left yet not from the earliest starting point of the schedule. Exactly once implies each server is gone by just once. It checks the current status of the server that is living in nearby memory of that server. It is actualized by making correlation at every server. Clone will move to each server however it never execute on ever server on the grounds that firstly it check the status of the server then it execute. In the event that the status of the server is faulty then clone will execute else it won't. Exactly once is a property of the fault tolerance.

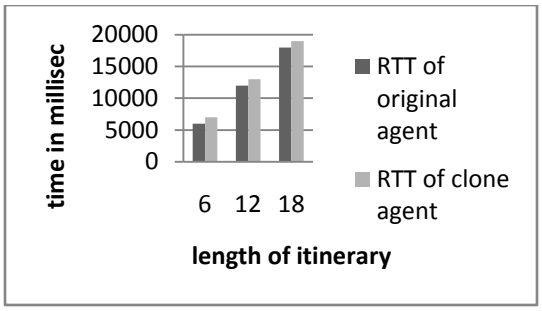
3 Results:

3.1 Evaluate Round Trip Time (RTT) Without Considering Fault.

Case 1 Without check pointing, round trip time is the time taken by the agent to successfully complete in its itinerary and return back to the home server. It is assume that an agent will take 1000ms to execute on the server and itinerary is 18. It show Rtt of original agent is less than the clone.

No. of host in an itinerary	6	12	18
Rtt of original agent	6000 ms	12000 ms	18000ms
Rtt of clone agent	7000 ms	13000 ms	19000ms

Table I Comparison of R_{tt} without Check pointing

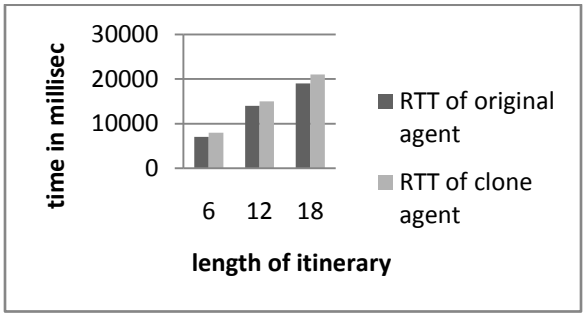


Comparison of R_{TT} Without Check pointing

Case 2 With check pointing, round trip time is reduces the time when the fault is occur in its itinerary.

No. of host in an itinerary	6	12	18
Rtt of original agent	7000 ms	14000 ms	20000 ms
Rtt of clone agent	8000 ms	15000 ms	21000 ms

Table II. Comparison of R_{tt} with Checkpointing



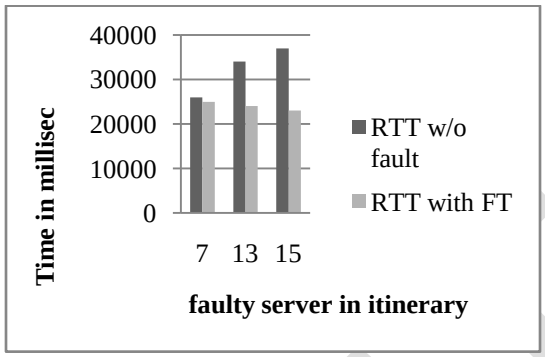
Comparison of R_{TT} with Checkpointing

Case 3: Evaluate the round trip time by considering the fault.This approach is deal with agent failure, cloning of the agent and check pointing is used. In Table III, Comparison of RTT by considering fault. It enhances the round trip time by limits the number of

server to visit again in case of fault tolerance and it is implemented after every 6th server.

Faulty server	7	13	15
RTT w/o FT	26000ms	34000ms	37000ms
RTT With FT	25000ms	24000ms	23000ms

Table III, Comparison of RTT by considering fault.

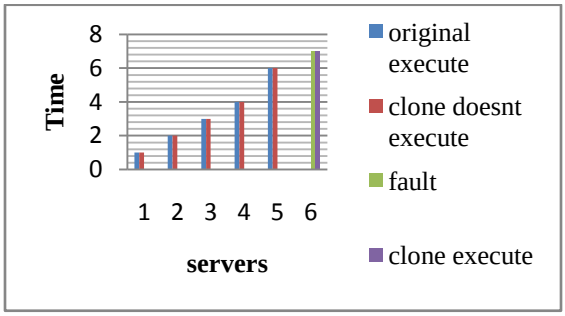


Comparison of RTT by Considering Fault

Case 4 The existing technique achieve the exactly once property, clone agent visits every server but it can execute only when the original agent is destroyed. The clone firstly checks the status then it execute.

	Read status of original agent stored in local memory	**Action performed to achieve exactly once**
Clone Agent arrives at server	Visited	Clone doesn't Executes
	Faulty	Clone Executes

Table IV. Comparison Mechanism Performed at Each Server



Exactly Once in Proposed Approach

4 Conclusions:

This methodology functions well with agent failure. To give fault tolerance, we utilized the concept of cloning the first agent so that the faulty unique agent can recuperate from its clone. Check pointing is utilized to utmost the rollback of repeated agent if there should be an occurrence of failure of both unique and clone agent. We are actualized the base paper in which it is material for both read just and additionally read/compose applications. In existing approach alongside the first agent its clone additionally visits every server except execute just for a situation when unique agent comes up short. After effective execution unique agent redesigns its status as gone to in nearby memory of particular server and moves to next server in its agenda. In the event that status of server is found as faulty at exactly that point clone executes else it moves to next server. Thus it protects exactly once execution of mobile agent and makes it suitable for both read just and read/compose applications.

5 References:

[1]. F. Yusuf, Z. Zaman (2009) "A Survey of Fault Tolerance Techniques in Mobile Agents and Mobile Agent Systems," In Proc. of International Conference on Environmental and Computer Science, pp: 454 – 458.

[2]. A. Budi, I. Alexei and R. Alexander (2006), "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 pp: 1-7.

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

[4]. M. Richa, H. Rahul, S. Gurpreet (2014) "A Novel Clone Based Comparison Approach for Fault Tolerance in Mobile Agent Systems," Proceedings of International Conference on Emerging Research in Computing, Information, Communication and Applications ERCICA pp: 243-248.

[5]. M. Richa, S. Gurpreet, K. Ramandeep and H. Rahul (2014), "A Review and Analysis of Various Fault Tolerance Mechanisms in Mobile Agents System," International Journal of Applied Research in Computing, Vol. 2, No. 1 pp: 343-350.

[6]. H. Rahul and K. Ramandeep (2012), "Novel Dynamic Shadow Approach for Fault Tolerance in Mobile Agent Systems," In Proc. of 6th International Conference on Signal Processing Communication Systems, Publication IEEE Conference pp:1-6.

[7]. A. Rostami, H. Rashidi and M.S.Zahraie (2010), "Fault Tolerance Mobile Agent System, Witness Agent in 2-Dimensional Mesh Network," In Proc. of International Journal of Computer Science Issues, Vol. 7, Issue 5 pp:153-158.

[8]. K. Mohammadi and H. Hamidi (2005), "Modeling of Fault-Tolerant Mobile Agents Execution in Distributed Systems," In Proc. of Systems Communications, Publication IEEE Conference.

[9] S. J. Choi, M. S. Baik, H. S. Kim, J. W. Yoon, J. G. Shon and C. S. Hwang(2004), "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.

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