Application of Mobile Agents as a Tool for Maintaining Consistency in Distributed Applications

Amit Mishra, Rajendra Purohit
Associate Professor
Jodhpur Institute of Engineering and Technology
Jodhpur

***Abstract*: - Distributed application executes on multiple nodes of remote sites. Due to involvement of multiple nodes, it requires error recovery algorithms. Traditional message passing techniques were proposed to design these error recovery techniques in past. These techniques generate heavy network traffic and other network overheads. The distributed system also suffers from the "Domino Effect", as it may roll back the transaction to its initial state. Here in this paper we are proposing mobile agents as a solution for error recovery in distributed environment as well as to nullify the domino effect. This new approach reduces the network traffic and provides most updated system information for decision making.**

I. INTRODUCTION

Here in this paper we are basically discussing the use of mobile agents as a tool for synchronization as well as for nullifying domino effect. First, we will discuss mobile agent and then we will discuss the domino effect.

Mobile agents- Mobile agents are moveable code and it contains executable code as well as some other information [1, 2]. Mobile agents are suitable for distributed environment because this new technology offers several advantages over existing old techniques. First, mobile agents can work independently and autonomously. Any specific task can associated with mobile agent and then it can be dispatched in the network. Now this agent becomes an independent entity in the network and it can work autonomously. Second, mobile agent technique reduces the network traffic as mobile agent can dispatch itself to any remote site where it can interact with that site and no packet movement is required for this interaction. Third, mobile agents contain executable code and it can sense the change in execution environment. After sensing it reacts to this change accordingly. So mobile agent can react dynamically to unpleasant situations and it provides better fault tolerance capabilities and robustness. So after considering these points, we can say that mobile agent technology is suitable for distributed applications.

Domino effect and Error recovery-

In a distributed system many users perform various operations and transactions concurrently. The ACID (Atomicity, Consistency, Isolation and Durability) properties of a transaction allow safe sharing of data. The mentioned properties can be achieved using recovery manager at each site of a distributed system and its main task is to save final change, due to transaction, in recovery file and restore the site to a consistent state when a failure occurs. It is known as Rollback in the system.

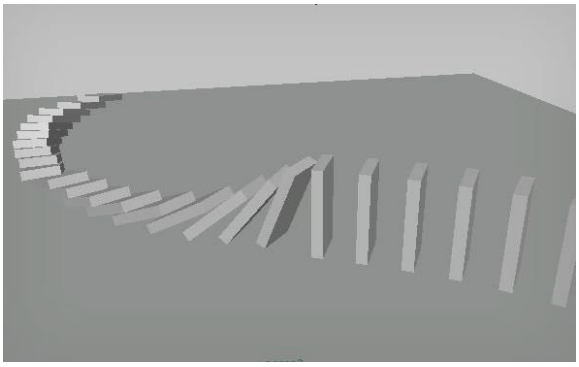

Fig.1 : Domino effect representation

In a distributed system these checkpoints must lead system a globally consistent state [3]. Checkpoints and error recovery algorithms can be used to maintain consistency in a distributed environment. A local checkpoint is a saved copy of an earlier local state of one process. A global check point is a collection of local check points, one for each system or process [4]. This approach may generate domino effect in the system. Domino effect can be represented using the Fig.1. We can better understand it using this example. Consider a situation where a sender of message *m* rolls back to a state that precedes the sending of m. The receiver of *m* must also roll back to a state that message precedes m's receipt; otherwise, the stats of the two processes would be inconsistent because they would show that message *m* was received without being sent. Under some scenarios, rollback propagation may extend back to the initial state of the computation, losing all the work performed before a failure. This situation is known as Domino effect in the system. It introduces the concept of mutually consistent processes or events. Two processes p1 and p2 are mutually consistent if:-

a) Every message recorded as "received from p1" in p2's state is recorded as "sent to p2" in p1's state and

b) Every message recorded as "received from p2" in p1's state is recorded as "sent to p1" in p2's state [10].
It can be represented using Fig1. as

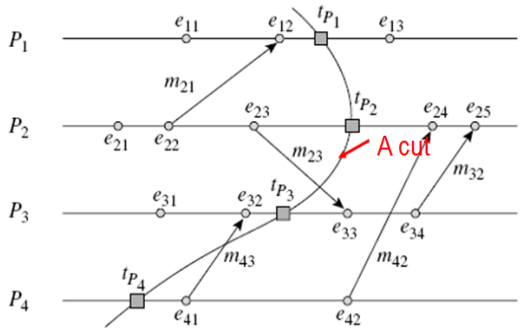

Fig. 2: Timing diagram for a distributed system with an inconsistent cut

The cut shown in Fig. 1 is not a consistent cut, as in this diagram, the send event of process p4 (e_{41}) is not recorded or saved as it is not a part of the cut. It is also creates orphan message. It is a message whose receiving event is recorded in the checkpoint, but its sending point is lost. In a distributed computing environment, this type of situation may lead our system to inconsistent state.

II. NEW APPROACH

In this paper we are combining the points discussed in the above section to maintain consistency and presenting this as a solution for error recovery. The proposed solution can be implemented using IBM's Aglets [9]. Mobile agents act as messengers or monitors that move from site to site over the network and coordinate the processes for checkpoints and rollback actions [4]. Traditional methods are expensive in terms of network overheads because multiple processes take part in the message sending and receiving activities. Each mobile agent can distribute itself in the network. Local checkpoint represents the recorded state of any machine or site. A global checkpoint can be defined as a set of all local checkpoints. In our approach, a group of mobile agents is used to control and monitor different conditions to coordinate consistent global checkpoints.

We can order local events using check point numbers called timestamps. Lamport or vector timestamps can be used for this purpose. These timestamps must maintain Happened-Before relationship. Domino effect may lead system to the initial stage of computation, as we discussed earlier [7]. To prevent this cascade rollback, we are using a special mobile agent named *Rollback coordinator*. It provides assurance that number of checkpoints will not exceed the predefined value. It works as follows: - If process X sends a message *m* to process Y, then the current timestamp and check point of X is added to the message header. Each process maintains

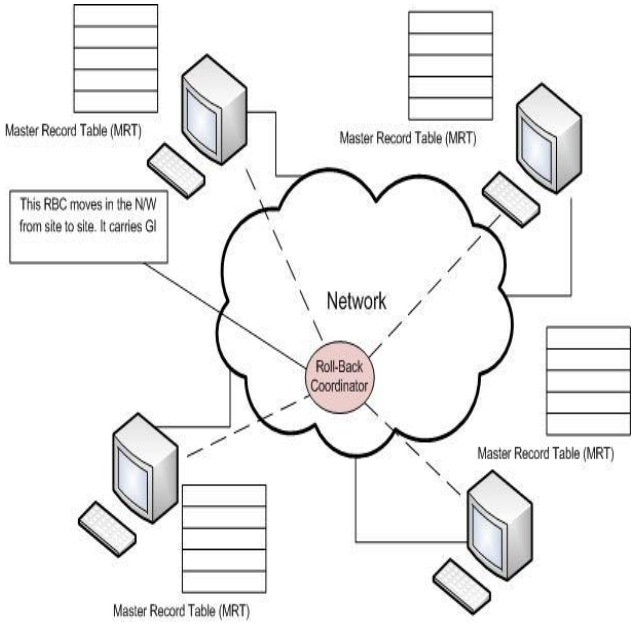

Fig.2 Use of mobile agent (RBC) for coordination

complete information of received messages by storing the header of each message in a record called *Message Record Table* (MRT).
Rollback coordinator moves from one site to another and it carries updated information of previously visited sites known as Global Information (GI)[4]. When it arrives at site X, it analyses the MRT of site X and then uses these records to update the GI and it identifies the dependencies between this site X and other sites. Now it calculates the number of rollbacks performed by this current site, if a fault is detected at this moment. If this count is greater than the predefined value, a coordinated checkpointing procedure must be performed to remove the possibility of domino effect [5,6]. Similarly it also calculates a new recovery line for rollback procedure. For this purpose, the Rollback coordinator generates a group of CC (consistency coordinator) agents, one for each site, and dispatches them to their corresponding sites. When this CC agent arrives at any other site, it monitors the local site to see whether there are any messages sent after last checkpoint. If this condition is true, it will force this site to for checkpoint procedure. This site is allowed to send any new messages only after this checkpoint procedure. It means the normal execution of this site is stopped after receiving this CC agent.
After this forced checkpoint procedure, this CC sends a *completion_message* to Rollback coordinator. After receiving this *completion_message* from all the sites, it sends *normal_op* message to all the sites to start the normal execution again.

III. CONCLUSION

As discussed above, the distributed system suffers from the domino effect. Due to this performance of system is also reduced. Traditional techniques of agreement in distributed system also generate heavy network traffic. The proposed

method solves this problem. It is efficient in terms of computation as well as time. Mobile agents can configure it according to network conditions and they can also take decisions to maintain consistency. Similarly mobile agents can also be used to nullify the effect of cascade roll-back operation.

REFERENCES

[1] Amit Mishra, Anamika Choudhary "Mobile Agent:Security Issues and Solution" International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 6, December 2012

[2] Hyacinth S. Nwana, "Software Agents: An Overview " Intelligent systems Research, Advanced Applications & Technology Department, Ipswich, Suffolk U.K. Cambridge University Press 1996

[3] Colin J. Fidge "Timestamps in Message-Passing Systems That Preserve the Partial Ordering" http://fileadmin.cs.lth.se/cs/Personal/Amr_Ergawy/dist-algos-papers/4.pdf

[4] Jiannong Cao, G.H. Chan, Weijia. Jia, Tharam S. Dillon "Checkpointing and Rollback of Wide-Area Distributed Applications Using Mobile Agents, "0-7695-0990-8/01 (C) 2001 IEEE

[5] E.N. Elnozahy, L. Alvisi, Y.M. Wang, and D.B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems", Technical Report CMU-CS-99-148, School of Computer Science, Carnegie Mellon University, October 1999.

[6] D.B. Johnson and W. Zwaenepoel, ``Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing", Journal of Algorithms, 11, 1990, pp462-491.

[7] D. Briatico, A. Giuffoletti and L. Simoncini, ``A Distributed Domino-Effect Free Recovery Algorithm", IEEE Proc. 4th Symposium on Reliability in Distributed Software and Database Systems, Oct. 1984, pp207-215.

[8] B. Bhargava, Shy-Renn Lian, ``Independent Checkpointing and Concurrent Rollback Recovery for Distributed Systems An Optimistic Approach", IEEE Proc. 7th Symp. on Reliability in Distributed Systems, Oct. 1988, pp3-12.

[9] C. Xu and D. Tao, "Building Distributed Applications with Aglet", http://www.cs.duke.edu/chong/aglet

[10] Randy Chow, T. Johnson "Distributed Operating Systems and Algorithm Analysis", Addison Wesley