International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Fault Tolerance in Mobile Computing using Multi Agent Environment for Electronic-Business Applications

Shiv Kumar[1], Shrawan Kumar[2]

[1]Mewar University, Chittorgargh, Department of Computer Science of Engineering, NH-79, Gangrar-312901, India

[2]Mewar University, Chittorgargh, Department of Computer Science of Engineering, NH-79, Gangrar-312901, India

Abstract: ***Mobile agent technology has become a new paradigm for distributed real-time systems because of their inherent advantages. In any distributed system, along with other issues, survivability and fault tolerance are vital issues for deploying mobile-agent systems. E-business becoming a prominent domain for deploying agent technology, it also faces reliability problems due to the failure of agent platform and communication link etc. The reliability is a factor that may affect the performance, availability, and strategy of mobile agent systems. Agent technology is a field of considerable active research. Various classes of agents (e.g. intelligent agents, software agents) have emerged so far. Software agents are useful for distributed systems & electronic commerce. However to fully deploy software agents in practice, a number of challenging issues especially security, fault tolerance and privacy need to be addressed. The scope of this discussion is limited to mobile agents in a multi agent environment for Electronic Business applications. Electronic commerce is one of the most important application areas of mobile agent technology. A secure mobile agent system (SMAS) model is established for e-commerce environment. E-Commerce and M-Commerce can help a company or enterprise to extend its market place to unlimited region***

Keywords: Mobile Agents, Trusted Mobile Agents in Multi Agent Environment, Security in E-commerce, Fault tolerance.

1. Introduction

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

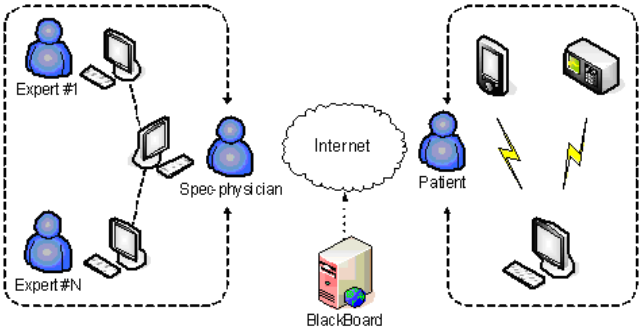
Fault-tolerant computing is the art and science of building computing systems that continue to operate satisfactorily in the presence of faults. A fault-tolerant system may be able to tolerate one or more fault-types including -- i) transient, intermittent or permanent hardware faults, ii) software and hardware design errors, iii) operator errors, or iv) externally induced upsets or physical damage. An extensive methodology has been developed in this field over the past thirty years, and a number of fault-tolerant machines have been developed -- most dealing with random hardware faults, while a smaller number deal with software, design and operator faults to varying degrees. A large amount of supporting research has been reported.

A mobile agent is defined as a class of agent with the ability during execution to migrate from one host to another where it can resume its execution and while this may assist in network traffic reduction and in overcoming latencies in the network, the ability of the agent to move around does however introduce significant security concerns." Mobile agents are no longer a theoretical issue since different architecture for their realisation have been proposed. The goal of mobile agent system is to provide a distributed computing infrastructure supporting applications whose components can move between different execution environments. Electronic Commerce can help a company or enterprise to extend its market place to unlimited region. But security is the primary concern for the E- Commerce application in mobile agent computing. The problem of security data section in a Mobile Agent from discovery & exploitation by a malicious host is a different task. Mobile agent systems provide a greater flexibility and customizability to distributed applications like Electronic Business & information retrieval in the current scenario. With the increasing market of Electronic Commerce it becomes an interesting aspect to use autonomous Mobile Agents for Electronic Business transactions.

1.1 Mobile Agent System Model:

The Object Management Group defines a software agent as "a computer program that acts autonomously on behalf of a person or organization". The following properties characterize agents:

- Pro- active (support of the user's work)
- Adaptive (learning the user's preferences or the ability to work on different platforms)
- Autonomous (limited communication with its creator)
- Intelligent (making 'intelligent' decisions)
- Mobile (can actively migrate in networks to different systems and move directly to the local resources, like databases or application servers)



Mobile agents are programs that can migrate from host to host in a network, at times and to places of their own choosing. The state of the running program is saved, transported to the new host, and restored, allowing the program to continue where it left off.

Mobile-agents are capable of continued, autonomous operation disconnected from the owner and they migrate to other hosts during their lifetime to perform their task. The use of mobile-agents saves bandwidth and permits off-line and autonomous execution in comparison to usual distributed systems based on message passing as shown in Figure 1 below. Essentially, a mobile-agent consists of code, data and state information needed to carry some computation [1].

1.2 Mobile Agent Fault Tolerance

Mobile agents react dynamically and autonomously to the changes in their environment, which makes them robust, and fault tolerant. They have the ability to distribute themselves in the network in such a way as to maintain the optimal configuration for solving the particular problem. If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch themselves and continue their operation on another host in the network [3]. Some important terminologies related to fault tolerance are as follows [4]:

- Fault
- Fault Models
- Error
- Failure
- Crash failure
- Omission failure
- Transient failure
- Byzantine failure
- Software failure
- Temporal failure

1.3 Mobile Agent Fault Tolerance in the Field of E-Commerce

In a scenario where a mobile agent is equipped with electronic commerce capabilities it necessary to guarantee that an agent does not get lost or duplicated during its itinerary. Thus, the main objective for the design of our architecture was to achieve a reliable, fault tolerant and secure agent Transfer which guarantees the exactly-once semantic (only-once-type-2-semantic).

E-Commerce can help a company or enterprise to extend its market place to unlimited region. At the same time, to let companies and enterprises can have transactions through Internet; more new techniques are developed for Internet and WWW applications. Agent technique is one of the important technologies developed to support the Internet applications [5].

The Architecture for e-marketplace: -The architecture for E-commerce interface is divided into three layers:

- Mobile Agent Platform
- Interface
- E-Commerce Platform

2. Role of Mobile Agent in Electronic Commerce Environment [2]

Mobile agents are well suited for electronic commerce. A commercial transaction may require real-time access to remote resources such as stock quotes and perhaps even agent-to-agent negotiation. Different agents will have different goals, and will implement and exercise different strategies to accomplish these goals. We envision agents that embody the intentions of their creators, and act and negotiate on their behalf. Mobile agent technology is a very appealing solution to this kind of problem. An electronic commerce transaction may be viewed in terms of four different phases [1]:

- Product brokering,
- Merchant brokering,
- Negotiation
- Payment and delivery

Product brokering consists in the gathering of information about the product that is going to be bought. Merchant brokering involves the evaluation of a set of alternatives in order to make the purchase. Making the decision implies considering all the tradeoffs that the various products offer: price, warranties, delivery time, and others. During the negotiation phase the agent settles the final terms of the commercial transaction. The characteristics of the market directly influence the outline of this phase. In markets where prices and characteristics are fixed, negotiation may not even exist. Finally, in the purchase and delivery phase of the transaction, the agent actually makes the acquisition and delivers the money (or its electronic equivalent) against the goods [6].

2.1 Fault Tolerance Thread Challenges

With e-commerce growing at a very high rate the need for mobile agents is also increasing. It has been proposed to use mobile agent technology to provide some of the services of e-commerce. Agents are sent out into the Internet by an interested user and gather the required information. If necessary they can collaborate with other agents, engaging in

information sharing, exchanging or buying. Schemes are revised to create marketplaces where agents can deal and even take part in auctions. For this purpose an agent is equipped with the mechanisms to deliver payment for purchased goods or services on behalf of its authority [7]. The following failure can be occur in this process:

A mobile agent crashes when its current local agent server halts execution, thus terminating all active mobile agents. Such an event is encountered when the host running the agent server platform crashes or a fault is encountered in the agent server process.

- No stable storage mechanism is provided at visited agent servers for the recovery of executing agents.
- Reliable communication links are assumed.
- All agent servers are correct and trustworthy.
- The home agent server is always available.
- At least once failure semantics are assumed, i.e. the agent performs its task at least once. If an agent server crashes the task is repeated at available agent servers. This assumption is applicable to application where mobile agents only consume information at agent servers.
- A mobile agent ignores crashed agent servers.
- A mobile agent consumes information at agent servers. The state of agent servers is not modified.

2.2 Problem and Solution Proposed

Based on above construction of mobile agent system, a paper brings forth the mobile agent-based e-commerce system framework. Its model is shown in figure 3.

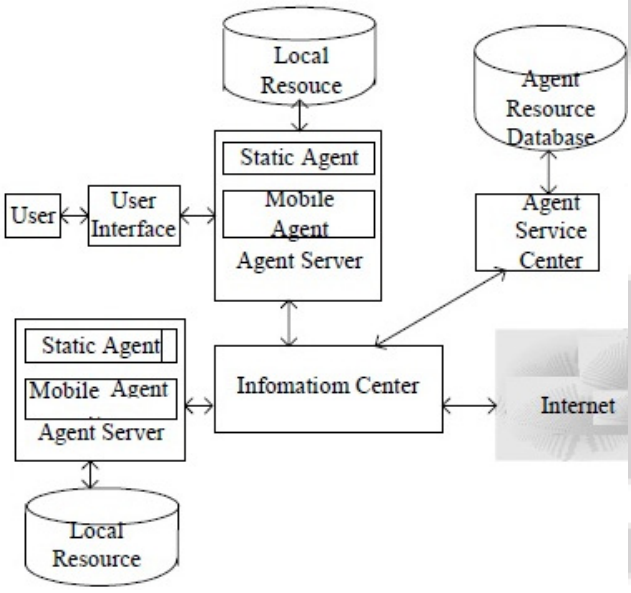


Figure 3: Mobile agent based E-Commerce system framework

In this system, agent service center provides catalog service, registers mobile agent's basic information, maintains agent's lifecycle, cancels illegal agent in time and provides agent query service. In order to enforce system security, especially for mobile agent protection, agent service center also maintains a table which records the history of host visitation, it gets task accomplish information of agent in visitation from its log files, calculates the host's credit degree, puts the result in the history table and re-calculates it so that mobile agent could optimizes routing strategy. Service center need to provide host service query function in order to help agent affirm the servers on which its task could be accomplished, and choose the best route. Following is the workflow of system:

1) User submits a request through user interface.
2) Agent Server analysis the request and creates static agent or mobile agent according to the request of task, then registers the agent's information in agent service center.
3) Static agent visits local resource under security control of agent server, completes user's request and returns the result to user. In the meantime, it registers agent's logout information on agent service center.
4) Mobile agent registers its information on agent service center, queries for host service table, returns agent server, works out migration route or plan according to routing strategy.
5) Mobile agent queries the history table of host visiting on agent service center, gets credit degree of host on the route, returns the optimized migration route of agent server.
6) Mobile agent sends out migration request to server, agent server makes agent hung up, encrypted, packed and sends it to destination host.
7) Destination host accepts agent, authenticates each other.
8) Destination host carries out security strategy according to authentication result, assigns limited execution environment for agent, and generates monitor agent to monitor the execution information of mobile agent.
9) Mobile agent carries out different security strategy according to authentication result, executes task under assigned resource. If necessary, it could make alliance with other agents and accomplish the task collaboratively.
10) Agent goes through hosts in routing table and completes its task, collects the results, returns it to user and submits log files to agent service center, registers logout information.

2.3 Multi-Version Software Fault Tolerance Techniques [4]:

Multi-version fault tolerance is based on the use of two or more versions (or "variants") of a piece of software, executed either in sequence or in parallel. The versions are used as alternatives (with a separate means of error detection), in pairs (to implement detection by replication checks) or in larger groups (to enable masking through voting).

Recovery Blocks: The Recovery Blocks technique combines the basics of the checkpoint and restart approach with multiple versions of a software component such that a different version is tried after an error is detected Figure 3). Checkpoints are created before a version executes. Checkpoints are needed to recover the state after a version fails to provide a valid operational starting point for the next version if an error is detected.

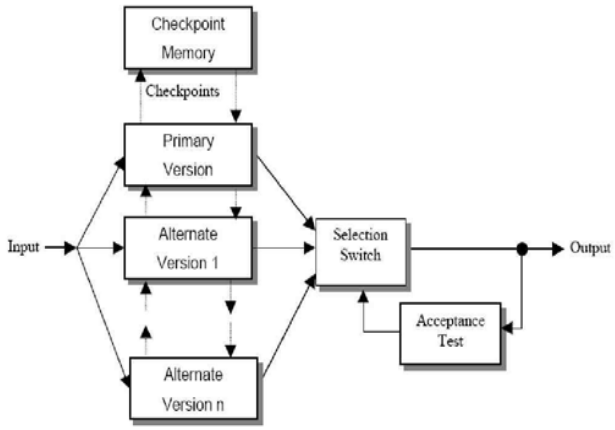


Figure 3: Recovery Block Model

N-Version Programming

N-Version programming is a multi-version technique in which all the versions are designed to satisfy the same basic requirements and the decision of output correctness is based on the comparison of all the outputs (Figure 4). The use of a generic decision algorithm (usually a voter) to select the correct output is the fundamental difference of this approach from the Recovery Blocks approach, which requires an application dependent acceptance test.

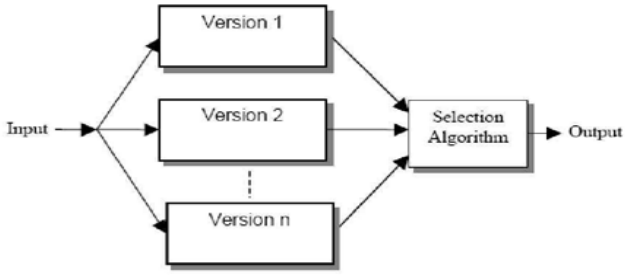


Figure 4: N-Version Programming Model

N Self-Checking Programming

N Self-Checking programming is the use of multiple software versions combined with structural variations of the Recovery Blocks and N-Version Programming. N Self-Checking programming using acceptance tests is shown on Figure 5. Here the versions and the acceptance tests are developed independently from common requirements. This use of separate acceptance tests for each version is the main difference of this N Self-Checking model from the Recovery Blocks approach

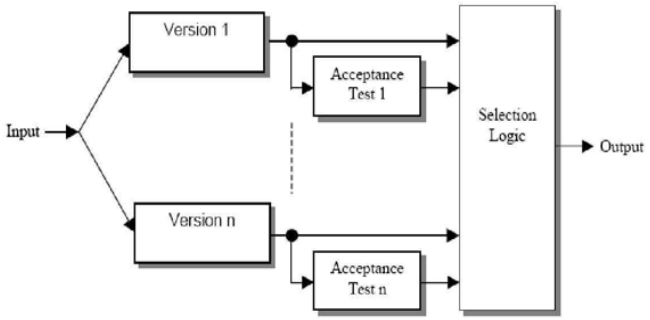


Figure 5: N Self-Checking Programming using Acceptance Tests

N self-checking programming using comparison for error detection is shown in Figure 6. Similar to N-Version Programming, this model has the advantage of using an application independent decision algorithm to select a correct output. This variation of self-checking programming has the theoretical vulnerability of encountering situations where multiple pairs pass their comparisons each with different outputs

Consensus Recovery Blocks

The Consensus Recovery Blocks (Figure 7) approach combines N-Version Programming and Recovery Blocks to improve the reliability over that achievable by using just one of the approaches. The acceptance tests in the Recovery Blocks suffer from lack of guidelines for their development and a general proneness to design faults due to the inherent difficulty in creating effective tests. The use of voters as in N-Version Programming may not be appropriate in all situations, especially when multiple correct outputs are possible. In that case a voter, for example, would declare a failure in selecting an appropriate output. Consensus Recovery Blocks uses a decision algorithm similar to N-Version Programming as a first layer of decision. If this first layer declares a failure, a second layer using acceptance tests similar to those used in the Recovery Blocks approach is invoked.

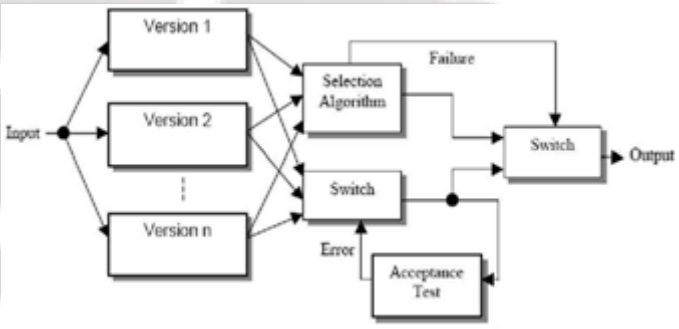


Figure 7: Consensus Recovery Blocks

Fault Tolerance Approach for enterprise applications [8]

Three classes of failures are identified in the literature:

- Behavioural and business logic failures,
- Operational failures
- Quality of service failures

It focuses on behavioural and quality of service failures. It introduces a model-based approach to failure management for Rich Service-based SOAS as follows. First they give a failure model, including the notion of a failure hypothesis. Then they introduce their approach to defining Detectors that identify occurrence of failures at run time. Finally, they introduce strategy-based Mitigators that provide recovery mechanisms after a failure is detected. As a consequence, we obtain the following benefits:

1) We can base the identification of a failure on the system model,
2) The logic to detect an error is separated from the logic to recover from it,
3) We can reuse the mitigation strategies in different contexts.

Here they consider mainly two types of failures: failures where a service does not complete its task, or where an error causes an unexpected message flow. Our approach can deal with both types of failures. We can detect unexpected message flow by comparing the messages exchanged with the sequences defined in the MSCs. Furthermore, our model allows the specification of deadlines between events enabling the detection of failures where messages are not sent. This capability is important in the web service domain to address the requirements of SLA. Deadline assertions can be leveraged to encode SLA, and detect possible violations.

In Mobile Agent-based Applications, a Survey

Al-Jaljouli et al [11] have implemented mobile agent in e-commerce to search and to filter information of interest from electronic markets. They describe also robust security techniques that ensure a sound security of information gathered throughout agent's itinerary against various security attacks, as well as truncation attacks. The figure 6 describes the sequence of processes carried out during the agent's lifetime. The authors utilize two co-operating agents where the initial verification terms are securely stored within a secondary agent (SA) that resides at the initiator and cooperates with a major agent (MA) that traverses the Internet.

Nipur et al. [12] propose a fault tolerant comparison internet shopping system Best Deal. The author has conducted the simulation by launching nine shopping mobile agents where each has to visit five supplier sites to get the best deal for different products. Performance is measured in terms of execution steps as well as execution time of the simulation. The mobile agent survives even if failure rate is more than 80% however for higher failure rate performance degraded significantly.

Li et al. [13] have studied mobile agent oriented M-commerce platform. The design and implementation of a mobile agent platform for M-commerce applications is discussed in this paper. According to the authors, the advantage of adopting mobile agents for M-commerce is to scale up to large, dynamic world market places distributed over the Internet and to ease the access and participation of mobile users

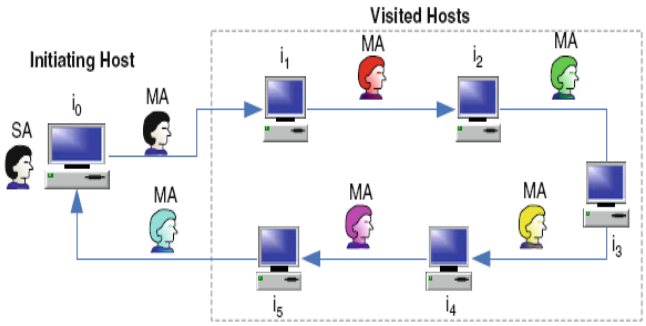


Figure 6: System architecture [11]

3. Drawbacks of Solution Proposed

Most of the software fault tolerance techniques have some advantages and also have some disadvantages.

Scientist "*Goutam Kumar Saha*" [9] inherits an overhead (of the order of three) of fault tolerance in both time and space. In order to tolerate potential transient faults. They cannot avoid such time and space redundancy. Though a self-stabilizing distributed algorithmic technique running on each node on the network is useful for bringing a system to a legal configuration in a finite number of steps from an arbitrary illegal system configuration caused by message corruption or sensor malfunction, it suffers from high time redundancy. Other fault tolerant schemes like, Algorithm Based Fault Tolerance in [18], Assertions in [19] etc., suffers from the lack of generality and wide applicability. Other costly schemes like, NVersion Programming (NVP) in [10], Triple Modular Redundancy (TMR) relies upon multiple versions of software and hardware. However, the proposed scheme is a low-cost solution because this relies on only one version of agent software, which is enhanced with the protective software fix. This approach does not lack in generality and applicability. This is intended to complement the intrinsic EDM in agent server systems towards tolerating random bit-errors. In other words, the proposed work is based on an enhanced single version-programming (ESVP) scheme using the extra protective code. The proposed technique needs three replicas or images of the agent code. It is assumed that an agent platform will allow its replication or reception. It does not need multiple independently developed versions of the agent-code. Rather it uses only an enhanced single – version (ESV) agent-code. It tolerates (detect & recover) one byte -fault in every three bytes at the same offset or displacement inside the three images of an agent.

4. Case Study

In order to better illustrate our approach in this section we present a survey on the topic "Fault Tolerance in Mobile Computing in Multi Agent Environment" as a running case study throughout the paper. The ability of a single agent is limited, so it is desirable for an individual agent to communicate and co-operate with other agents to perform complex tasks that are beyond its own capability. Fault tolerance is fundamental to further development of mobile

agent based applications. Fault tolerance is the ability of a system to perform its function correctly even in the presence of internal faults. Based on duration, fault can be classified as transient or permanent. A transient fault is not reproducible because it will eventually disappear, whereas a permanent one will remain unless it is removed by some external agency. A particularly problematic type of transient fault is the intermittent fault that recurs, often unpredictably. General fault tolerance procedure includes error detection and error recovery. Error detection is the process of identifying that the system is in an invalid state. We simulate the behaviour of an E-commerce in terms of fault tolerance for a multi agent environment.

5. Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Mr. Shrawan Kumar Sharma for the continuous support of my study, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this paper. I could not have imagined having a better advisor and mentor for my M.Tech study.

Besides my advisor, I would like to thank the rest of my department professors or lecturers: Prof. R.P.Ojha and Santosh Upadhyay and others, for their encouragement, insightful comments, and hard questions. Last but not the least; I would like to thank beloved, Sneha Rani supporting me spiritually throughout my life.

References

[1] Research on Mobile Agent-based E-Commerce System Framework Wenna Liu1, Deli Yang2 School of Management, Dalian University of Technology, Dalian 116024, China1 wenal@sina.com, 2 somdyang@dlut.edu.cn
[2] Reliable Multi Agent System For E-Business Applications A. Kannammal*, V. Ramachandran**, N.Ch.S.N. Iyengar*** *Dept of Computer Technology & Applications* Coimbatore Institute of Tech. Coimbatore 641 014 *kannaphd@yahoo.co.in*
[3] ***Dept of Computer Science, College of Engineering, Guindy Anna University, Madras 600 02 rama@annauniv.edu*
*[4] ***School of Computing Sciences, Vellore Institute of Technology,Vellore-632014 nchsniyr@yahoo.com*
[5] Mobile Agents Intelligent Assistants on the Internet
[6] Analysis of Different Software Fault Tolerance Techniques Golam Moktader Nayeem, Lecturer, Department of ECE, Southern
[7] University Bangladesh & Mohammad Jahangir Alam, Lecturer, Department of CSIT, Southern University Bangladesh
[8] W. Jansen. Counter measures for Mobile Agent Security. In Computer Communications, Special Issue on Advances in Research and Application of Network Security, November 2000. http://citeseer.ist.psu.edu/article/jansen00countermeasures.html
[9] Research on Mobile Agent-based E-Commerce System Framework Wenna Liu1, Deli Yang2 School of Management, Dalian University of Technology, Dalian 116024, China 1 wenal@sina.com, 2 somdyang@dlut.edu.cn
[10] Analysis of Different Software Fault Tolerance Techniques Golam Moktader Nayeem, Lecturer, Department of ECE, Southern
[11] University Bangladesh & Mohammad Jahangir Alam, Lecturer, Department of CSIT, Southern University Bangladesh
[12] International Computer Software and Applications Conference, 2009, p. 50-45.
[13] A Fault Tolerance Approach for Enterprise Applications Vina Ermagan, Ingolf Krüger, Massimiliano Menarini *University of California San* Diego {vermagan, ikrueger, mmenarini}@ucsd.edu.
[14] Goutam Kumar Saha ,Transient Fault Tolerance in Mobile Agent Based Computing Scientist-F, CDAC, Kolkata, India, Mailing Address: CA – 2 / 4 B, Baguiati,Deshbandhu Nagar, Kolkata 700059, West Bengal, India, gksaha@rediffmail.com
[15] Avizienis, A. The N-Version Approach to Fault Tolerant System, IEEE Trans. Software Engineering, v. SE-11, n.12, p.1491-1501, 1985.

Author Profile

Shiv Kumar received the M. Tech. degree in Computer Science and Engineering from Mewar University Chittorgargh in 2012. During 2007-2013, he stayed in Canon India Private limited Center of Excellence center and India Software Center Noida and Gurgaon of India. He know with Mewar University, Chittorgargh.

Shrawan Kumar Sharma is currently pursuing master's degree program in Computer science and engineering in Mewar University, India,