American Journal of Software Engineering and Applications
2013; 2(5): 111-124
Published online September 20, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20130205.11

A systematic review of fault tolerance in mobile agents

Bassey Echeng Isong[1], Eyaye Bekele[2]

[1]Deptmemt of Computer Science, University of Venda, Thohoyandou, South Africa
[2]School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden

Email address:
bassey.isong@univen.ac.za(B. Isong), eyayeb@gmail.com (E. Bekele)

To cite this article:
Bassey Echeng Isong, Eyaye Bekele. A Systematic Review of Fault Tolerance in Mobile Agents. *American Journal of Software Engineering and Applications*. Vol. 2, No. 5, 2013, pp. 111-124. doi: 10.11648/j.ajsea.20130205.11

Abstract: Mobile agents have engrossed substantial attention in recent years, especially in fault tolerance researches and several approaches have emerged. Fault tolerance design tends to put a stop to incomplete or complete loss of the agent in the face of failures. Despite these developments, reliability issues still remain a critical challenge. Moreover, there is no comprehensive detail bringing together, summaries of the existing efforts of researches in order to focus attention where it is needed most. Therefore, our objective in this systematic literature review (SR) is to explore and analyze the existing fault tolerance implementations in order to bring about the state-of-the-art and the challenges in mobile agent's fault tolerance approaches. We used studies from a number of relevant article sources, and our results showed the existence of twenty six articles. Our analysis indicates that the existing approaches are not generic and each focuses on a specific aspect of the problem, usually in one or two specific fault models which impacts on agent's reliability. The implication of the study is to give a clear direction to future researchers in this area for a better reliable and transparent fault tolerance in mobile agents.

Keywords: Mobile Agents, Fault Tolerance, Replication, Check-Pointing, Systematic Review, Platforms

1. Introduction

In the last few decades the field of mobile agents in distributed computing has witnessed substantial attention in both academia and industrial fields. This is known to be stimulated by the exponential growth of the Internet and system dependability. However, in spite of all, mobile agent's reliability is still a critical issue. Due to their nature, mobile agent's reliability and execution is not failure-free in the environments they operate. The growth of distributed heterogeneous environments such as the Internet naturally exposes them to abnormal conditions originating from migration request failure, communication exceptions or security violation [3],[4],[6]. Hence, providing reliability is of the essence to integrate agent-driven systems into today's industrial applications. Mobile agents have to be made reliable through fault tolerance to withstand adverse environmental situations in today's industrial applications. Fault tolerance is designed to provide reliable execution of mobile agents even in the presence of system failure.

Achieving mobile agent's fault tolerance requires the adherence to the non-blocking and exactly-once execution [7]. Presently in literature, several mobile agents' fault tolerance approaches exist in variety of mobile agent platforms. These approaches used different mechanisms to provide the reliability mobile agents' execution needs especially in the failure detection and recovery aspects. Moreover, majority of the recent approaches are based on optimizations, hybrid-based, while others are based on exception handling. In general, the existing fault tolerance schemes are categorized as either curative in nature (e.g. exception handling) or preventive [8], [9], [10]. The preventive schemes are further categorized into check-pointing and replication-based schemes but in some cases a hybrid of both schemes [8].

Despite several efforts and interest in this field, there is no comprehensive detail bringing together, summaries of these efforts. The main gap in this research area lies in the fact that the existing approaches are not generic and each focuses on a specific aspect of the problem, usually in one or two specific fault modes which is known to have huge impacts on agent's reliability. To improve the reliability of a system, faults originating from different forms need to be addressed in the fault tolerance measure. Unfortunately, there is no fault tolerance framework in the existing literature that serves as a guideline for realizing the state-of-the-art in mobile agent's fault tolerance. Therefore, bringing together fault tolerance approaches in mobile agent will assist researchers in closing the gaps identified in this study. The objective of this systematic literature review (SR) is to explore the existing fault tolerance approaches in mobile agents system to identify the current state of research, the techniques and approaches used, and the factors that influence the execution reported in recognized fault tolerance implementations research.

The rest of this article is organized as follows. An overview of mobile agents is given in Section 2. The research methodology of this SR is described in Section 3. The analysis of the results in accordance with the research questions is presented in Section 4. Section 5 provides the study discussions while the conclusions and recommendations are in Section 6.

2. Overview of Mobile Agents

Mobile agents are encapsulated pieces of executing program that have the ability to travel from one host to another and perform certain task autonomously [3]. It is a technology that aimed at shifting computation towards the data other than the other way round [1]. Mobile agents have characteristics that are distinct, thus making them flexible in deployment and desirable for use in distributed applications than other technological paradigms such as client-server, peer-to-peer, and others [2]. (see Figure 1.) These features include ability to naturally operate in heterogeneous environments [9], act autonomously [8], move independently from one host and can effectively make execution decision 14]. Mobile agents heavily rely on the underlying protocol for communications by way of interactions and message exchanges in order to successfully carry out and execute certain task in the in the agent system [9].

Based on their characteristics, mobile agents provide several benefits such as bandwidth conservation [15], asynchronous and autonomous interactions [8], extended flexibility in disconnected data operations [16] and can improve network latency with better response time [16], robust and fault tolerant [17]. Mobile agents are generally independent of the computer-layer and transport-layer but dependent only on their execution environment [18], and have better scalability [16].

Today, the concept of mobile agents is receiving considerable attention in both research and industrial fields. Several platforms exist that provides operating environments for mobile agents such as Aglets, Agent Tcl, Knowbots, Telescript, Voyager, Mole, Tacoma, Grasshopper, James, Swarm and others [1], [2], [13]. Moreover, they have applications in several areas such as e-commerce and m-commerce, network monitoring and management, distributed information retrieval, telecommunications, remote device control and configuration, Internet computing, etc [2],[5],[8],[19],[20],[22]. To this end, despite the flexibility offered by mobile agents, agents are not isolated from several challenges such as malicious or errant hosts, erratic Internet behaviors or resource scarcity [6]. These therefore, calls for reliability and security mechanisms to be in place [9],[12]. The reliability issue is being addressed by fault tolerance mechanisms, which is the focus of this study.

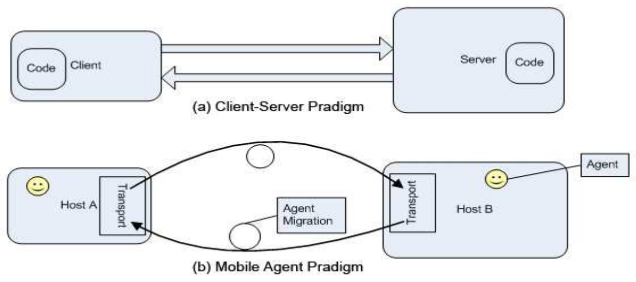

Figure 1. *Client-server vs. mobile agent paradigm*

3. Research Method

Systematic review is a methodology aimed at minimizing the inconsistencies associated with less scientifically rigorous review methodologies through strict qualitative research methods resulting to objective and unbiased results. In this study, we have applied SR to explore the state-of-the-art of fault tolerance in mobile agents by following the guidelines in [23]. The steps involves are discussed in subsequent sections.

3.1. Research Questions

This SR aimed to summarize the existing mobile agent's fault tolerance approaches in recent years. It would provide a list of reported and recognized techniques and approaches, influencing factors, platform supports and challenges in mobile agent's fault tolerance. Therefore, the research questions are:

SRQ1. What is the state-of-the-art in research of the recognized mobile agent's fault tolerance?

SRQ2. What available fault models are considered in designing fault tolerance protocols?

SRQ3. What are the available approaches and their design elements in the available recognized mobile agent's fault tolerance in current state of research?

SRQ4. What factors influences mobile agent's fault tolerance execution?

SRQ5. How much supports are offered by the mobile agent's platforms used in implementing the fault tolerance features?

SRQ6. What challenges exist and how do they affects the implementation of the fault tolerance in mobile agents?

3.2. Search Strategy

A search strategy is designed to ensure that all relevant studies other than irrelevant ones appear in the search result. In this SR, our literature search is limited over the scopes of: publication time period and publications that discusses fault tolerance in mobile agents. We considers the review of 10-years' efforts in mobile agents fault tolerance, that spanned from January 1998 to December 2008. We selected these periods in order to obtain relevant and sufficient information that are of the essence to this study and provide evidences of the trends in mobile agent fault tolerance then.

Therefore, any paper published after 31 December, 2008 is not included in our search result. We limited our searches to the electronic databases: Compendex/Inspec, IEEE Xplorer, Google Scholar, ACM digital library, Springer link and Scirus since they contain peer-reviewed works published in journals, digital libraries, conferences, proceedings and workshops which are of recognized quality within the software engineering research community. In this study, the quality of each selected research article was evaluated against a number of checklist questions. Each of questions is answered based on three options along assigned weights: Yes=1, Partial=0.5 and No=0. The maximum score a particular publication can get is 8.(see Table 1)

Table 1. *List of selected publications by publisher and methodology*

Ref.	Authors	Year	Publisher	Published in	Methodology	Quality Score
[25]	Summiya	2006	IEEE	Conference	Model and Simulation	4.5
[26]	Leung, Kwai Ki	2005	IEEE	Conference	Model and Simulation	6
[14]	Kyeongmo Park	2004	Springer	Conference	Model and Experiment	5
[5]	Guiyue Jin	2004	Springer	Conference	Model and Simulation	5
[27]	Lyu, M.R.	2003	IIIS	Conference	All	5.5
[28]	Sehl Mellouli	2007	Springer	Conference	Model and Experiment	4.5
[29]	Assis, Silva, F.M.;	1998	Springer	Workshop	Model	3
[30]	Osman, Taha	2004	IEEE	Journal	Model	3
[31]	Meng, Xuejun	2006	IEEE	Conference	Model and Experiment	5
[32]	Jong-Shin Chen	2008	IEEE	Conference	Model and Simulation	3
[33]	Marin, Olivier	2005	Springer	Workshop	Model and Experiment	8
[34]	Lyu, Michael R.	2004	IEEE	Journal	Model and Simulation	4.5
[35]	Youhei Tanaka	2006	IEEE	Conference	Model and Simulation	3
[36]	Silva, Luís Moura	2000	IEEE	Conference	Model and Experiment	6
[7]	Pleisch, Stefan	2003	IEEE	Journal	Model and Experiment	7
[38]	Rothermel, Kurt	1998	IEEE	Conference	Model	3.5
[39]	Alan Fedoruk	2002	ACM	Conference	Model and Experiment	6.5
[40]	Taesoon Park	2004	Springer	Conference	Model and Experiment	5.5
[41]	Taesoon Park	2004	Springer	Conference	Model and Experiment	6
[42]	Mohammadi, K.	2005	IEEE	Conference	All	5
[44]	Yang, Jin	2005	Springer	Conference	Model and Simulation	5
[46]	Park, Taesoon	2004	Springer	Conference	Model and Experiment	5
[47]	Tomoaki Kaneda	2005	ACM	Conference	Model and Experiment	5
[48]	Park, Taesoon	2006	Springer	Conference	Model and Simulation	3
[43]	Johansen, D.	1999	IEEE	Conference	Model and Experiment	5.5
[45]	Milovan Tosic	2005	Springer	Conference	Model and Experiment	4.5

The study selection process was individually carried out by the authors involved and any differences were settled by consensus. The search strategies we adopted were iterative in nature and the inclusion/exclusion decisions were checked at least twice and discussed at each stage of execution. We adopted a multi-stage process in selecting the studies in accordance with the guidelines in [23], using different selection criteria. All search terms we created were applied on the selected databases and a total of 6,901 results were found. In the first stage, 6788 articles were excluded based on the relevance of their title or abstract. Furthermore, the titles and abstract of the left over 113 were read and the basic inclusion and exclusion criteria applied, leaving a total of 86 studies. In the second stage, 55 out of the 86 articles were selected based on the application of the detailed inclusion and exclusion criteria - abstract, the conclusion and in some cases the introduction was reviewed to apply the exclusion criteria. The exclusion criteria were based on inaccessibility, formal or mathematical description, and exception handling. Lastly, in the third stage, we based the selection process on the detailed research questions while the exclusion criteria was based on issues of duplication, application of mobile agents in a different study area, and mobile agent's platforms articles. As a result, 26 unique studies were selected as primary studies for this SR and 29 studies were discarded.

3.3. Data Extraction and Synthesis

The data extraction strategy was developed in accordance with the research questions, quality assessment checklist, general information associated with the study identification and certain common characteristics in the studies. During the extraction, the authors also checked and re-checked the extracted data to get rid of uncertainties. To assist us find and validate the extracted information and resolve inconsistencies quickly, we tinted all important lines and paragraphs in the selected studies. Accordingly, difficulties encountered were resolved via discussion among the authors. For multiple articles of the same information, articles with the most complete and latest information were used to avoid unbiased findings. To extract relevant information, we created and used data extraction form with the following fields: Title, Authors names, Journal/Conference/Workshop, Year, Research Methodology, Moble agent fault tolerance Scheme, Protocol, Fault model, Assumptions Detection, Recovery, Fault Tolerance execution, Agent types, Communication, Factors affecting the performance of the proposed model and experiment variables, Platform type, Platform support and Challenges.

With the extracted data from the extraction forms coupled with the nature of this study, we performed descriptive synthesis of the data since it is the only suitable method in such heterogeneous data format.

4. Analysis

In this section, we present analysis of the results of this SR by answering the research questions as follows:

4.1. Mobile Agents Fault Tolerance Research

RQ1: What is the state-of-the-art in research of recognized mobile agent's fault tolerance?

To answer this research question, analysis will be based on publication years, the qualities of the articles and the methodology used. Table 1 presents list of selected publications by publisher and Methodology as well as the quality score for each study.

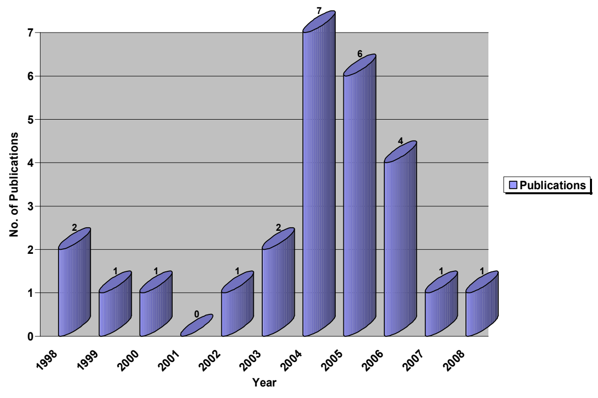

Figure 2. *Studies by year of publication.*

4.1.1. Year of Publications

We found 26 studies related to mobile agent fault tolerance where 21 studies were published in conference proceedings, 2 studies in Journals and 3 studies in workshops. Analysis shows that the field of mobile agent fault tolerance was active in research in those periods. Further analysis indicates that years between 2004 and 2006 have showed a remarkable increase in number of publications, though the trend seems to be going down in the last three years (2006 - 2008). Figure 2 and Table 1 show studies by year of publication.

4.1.2. Publication Quality Scores

Analysis shows that more than 75% of the selected publications scored 4.5 or more. Articles with more than 4 as score in the quality assessment generally are selected on the basis of having the most vital information such as detailed description of a model and some form of proof such as results from a real experiment or simulation to support their findings. (see Figure 3)

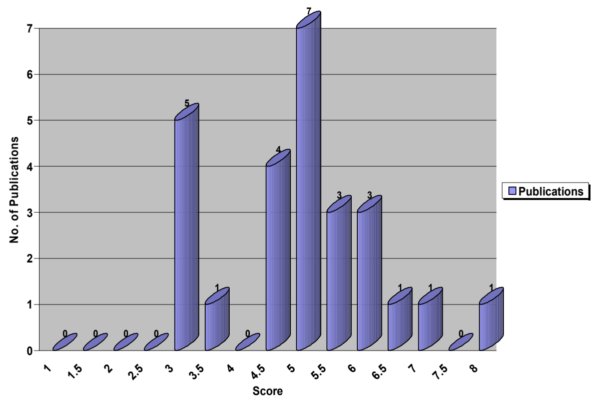

Figure 3. *Publications by quality assessment scores*

4.1.2. Methodology

In this study, we noticed that the number of studies on fault tolerance in mobile agents that were supported by simulation or experiment has improved over the years. Analysis indicates that about 46% studies are with experiments, 31% is simulation only, while a combination of experiments and simulations has 8%. Only 12% of the studies discussed fault tolerant models with no experiments or simulations. (see Figure 4).

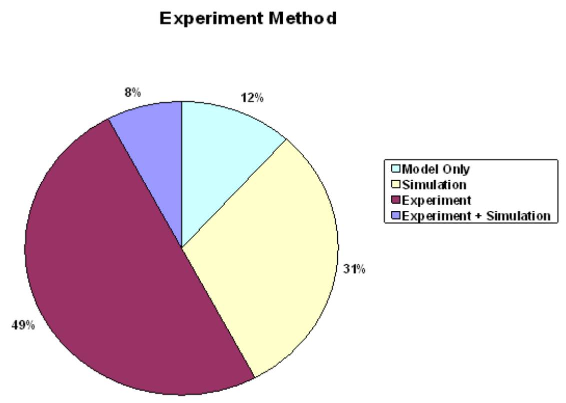

Figure 4. *Primary studies research methodology*

4.2. Fault Models

SRQ2. What available fault models are considered in designing fault tolerance protocols?

In the 26 studies, we found that the ability to observe the agents and detect failure during execution is one of the cardinal features of any fault tolerance in mobile agent approach. This is because mobile agents or its environments are not failure-free. We noticed '*fault model*' is used to define which set of observations are categorized as failure and which are acceptable operation modes. Analysis shows that all the existing approaches have fault models and there are three classes of agent's failures: *communication, crash* and *agent software* failure. Moreover, we found that most of the studies are designed to either cater for one of the stated failures or multiple of them. Table 2 presents the existing implementations of each study and the failure types designed for them.

Table 2. *Failure Types*

Failure Types		
Communication	**Crash**	**Agent/Agent Software**
[25],[30],[31],[32], [42],[36],[38],[39], [43], [44], [45]	[5],[7],[25],[26],[27], [29],[30],[31],[32], [34],[42],[35],[36], [38],[40],[41],[14], [44],[46],[47],[43], [45]	[25], [26], [5], [27], [28], 29],[30],[31],[32], [33], [42], [7], [36], [38], [39], [40], [41], [44], [46], [47], [45],[48]

Based on Table 2, the distribution of fault models in the studies presented in Figure 5 denotes that fault models for communication failures were least addressed (43%). This could depend on the assumption that the network is reliable or agents can eventually resume service even if the network fails. Crash and agent software fault models seem to be supported in most of the existing implementations (84%). Only about 35% of the studies supported all three fault models [44], [27], [38], [30], [31], [32], [42], [36], [45], while 12% of these studies considered the issue of network partitioning and suggested a solution to it [44], [38],[30].

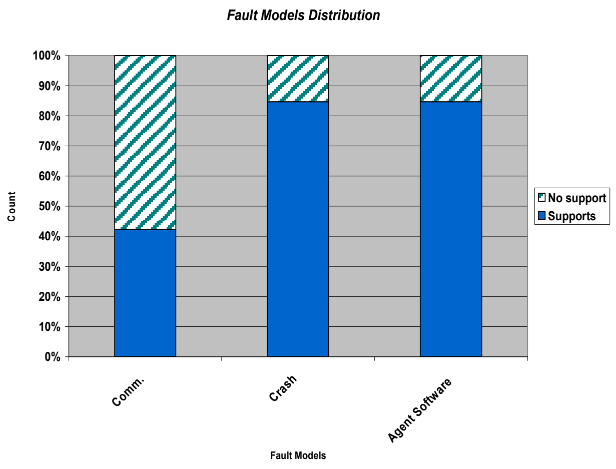

Figure 5. *Fault Model distribution*

4.3. Mobile Agents Fault Tolerance Approaches and Design

SRQ3. What are the available approaches and their design elements in the available recognized mobile agent's fault tolerance in current state of research?

This research questions will be answered based on the different approaches of mobile agent's fault tolerance, the available protocols and the design of the available implementations.

4.3.1. Fault Tolerance Schemes

One of the key features of fault tolerance approach is the ability to recover from failure. In this study, we found that the existing fault tolerance schemes that deal with sources of system failures and recovery were categorized as either the replication-based, checkpoint-based or a hybrid schemes [16], [9], [49]. In this study, we only considered the two most widely used schemes: check-pointing and replication for analysis. The different studies that used these approaches in the perspective of transactional and non-transactional approach are shown in Table 3.

Table 3. *Fault tolerance schemes and execution modes*

Agent's Execution Modes		
Scheme	Transactional	Non-transactional
Checkpoint	[5], [38], [36], [27], [30], [26], [42]	[34], [45], [48], [46], [32]
Replication	[29], [39], [7], [47], [35]	[43], [40], [41], [33], [44], [25], [28]
Hybrid	[14], [31]	-------

Further analysis shows that both check-pointing and replication-based schemes are used almost equally over the years of consideration (see Figure 6).

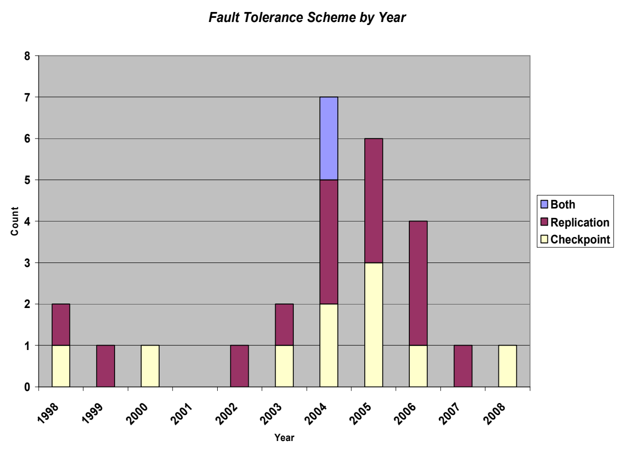

Figure 6. *Fault tolerance scheme distribution*

4.3.2. Replication Scheme Types

For replication-based schemes: active and passive, analysis in this study shows that the trend of distancing away from active replication-based schemes is quite evident. (See Figure 7) In addition, about 72% of the replication-based approaches used either the passive or semi-active replication, but the semi-active to passive ratio within the passive replication-based category is 3:7. This indicates that semi-active techniques are not used often either. It could be as a consequence of high computation and communication cost they incur. The computation overhead in active replication is higher even among the implementations that support both replication types.

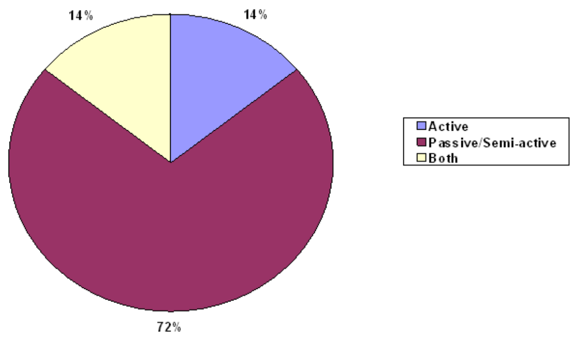

Figure 7. *Replication scheme types distribution*

4.3.3. Execution Modes

We found that execution mode in existing agents fault tolerance are either transactional or non-transactional. From all indication, we noticed that the distribution of the transactional and non-transactional executions is someway balanced (see Figure 8), though the transaction-based executions are slightly higher with about 54% than non-transaction-based modes with 46%. The analysis indicates there is no noticeable shift in trend over the years. However, transactional executions are more reliable in maintaining the exactly-once property of agent execution, while its counterpart maintain lower computation overhead. But we recommend transactional execution for application domains that require higher level of consistency.

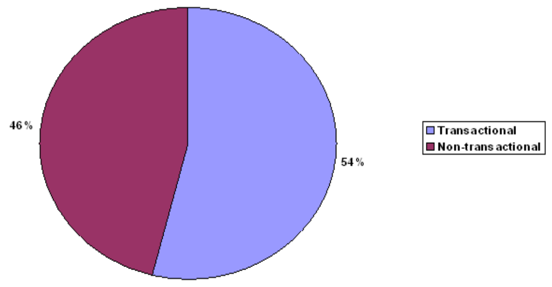

Figure 8. *Agent's execution modes*

4.3.4. Communication Modes

In this study, we found that there are only two communication modes in fault tolerant mobile agent dominated by the asynchronous mode. Analysis shows that about 92% of the studies implemented asynchronous mode execution while only 8% of the studies are both synchronously and asynchronously. (see Figure 9) There was no implementation that solely works on synchronous mode. We believe this could be as a result of the characteristics of the agent's environment where they are autonomous and migrate usually in open networks with latency. In addition, there is high performance overhead with synchronous when compared to asynchronous.

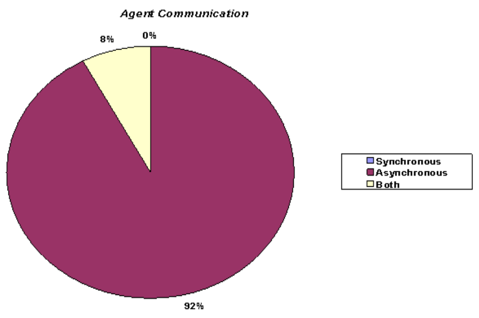

Figure 9. *Agent's communication mode distribution*

4.3.5. Fault tolerance Protocols

In this SR, we found that protocols in mobile agents fault tolerance implementation model are mostly achieved through effective message passing to coordinate and ensure reliable agent failure detection and recovery. In addition, categorizing these protocols into classes is not easy since some of them exist in association with other protocols. However, they can only be organized with respect to the two execution properties of mobile agents: the exactly-once or non-blocking. Figure 10 shows the class of existing protocols in mobile agents fault tolerance implementation model.

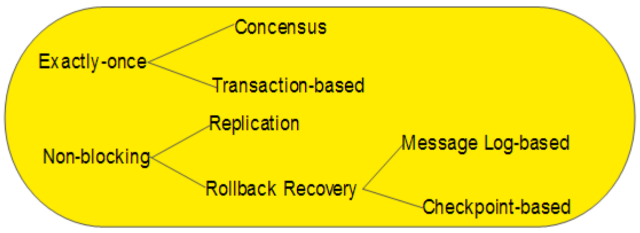

Figure 10. *Agent's proptocols*

4.3.5.1. Exactly-onec Protocols

This protocol guarantees that an agent must execute the desired action not more and not less than once in a host. This protocol includes the consensus and the transaction-based approaches.

A. Consensus Protocol: This protocol provides a means of achieving agreement between the primary agent process and the replicas in the case of failure. This is usually achieved by voting on a certain result by all the participating agents where a consensus is reached when all the participants agree on one outcome. In this case, a replica can become the new primary through its own consensus for either being the first to detect or having the highest priority when the primary fails.

B. Transaction-Based Protocol: Similar to consensus, but uses the transaction commits/abort to reduce the effect of failures on the availability of operational sites during distributed transaction. Commit and abort are both irreversible and unlike consensus, transaction can only be committed if all participating sites vote or agree to commit, otherwise, the transaction is aborted. In this study, analysis shows that most of the studies used the basic transaction-commit protocol such as centralized transaction commit by [26] while [7],[27],[30],[35],[36],[47] used the transaction commit protocol. A hybrid approach of both consensus and transaction-based protocol were implemented by [29],[47],[7],[38].

4.3.5.2. Non-Blocking Protocols

These protocols are used basically for overcoming blocking problems pose by the exactly-once protocols. It is achieved by either restarting a failed process or use duplicate agents that are able to take over in time of primary agent failure without voting. Protocols under this category include the rollback recovery protocols and the replication-based protocols.

A. Rollback Recovery Protocol: This approach tries to reduce the amount of loss in computation by avoiding the restarting of computation from the beginning. It provides a technique that requires a process to periodically record its consistent state, known as check-pointing, into a stable storage. Most of the existing rollback recovery approaches are based on message passing: checkpoint-based and message logging-based.

1. Checkpoint-based: This approach depends strictly on regularly saving agent's process states and code to a stable storage for future restoration or recovery in the event of failure. Protocols in this category include coordinated, uncoordinated, communication-induced, lazy and the timer-based protocol. In this study we found that most of the rollback-based protocols used this protocol to provide fault tolerance behavior such as communication-induced check-pointing [42] independent check-pointing with receiver-based logging [46], checkpoint-based scheme to restore agents processes back to its consistent state during failure, and checkpoint-based with reliable publisher/subscriber messaging layer[45].

2. Message Log-base: This is the logging of non-deterministic actions preset as determinants in combination with check-pointing to achieve fault tolerance recovery behavior. Protocols under this category include the pessimistic and optimistic logging protocol [7]. Pessimistic execution ensures that changes are applied only if no agent crashes and there is no erroneous result, while in optimistic execution the place modifications can be immediate and transparent but undoing modifications is a complex task [7].

B. Replication-based Protocols: In this approach, there is a live backup agent, either as a duplicate of the primary or as a standby, even before failure is detected. One drawback is that it requires synchronization of replicas with the primary, which is very expensive both in computation and communication. However, several optimizations have been suggested in the studies we considered such as formed a group of replicas and a proxy that communicates with the primary on behalf of the multiple replicas by [39], dynamic adaptive replication scheme by [33] and the sliding window protocol by [25], a technique that controls the number of backup or replicated agents in order to minimize bandwidth consumption.

3.3.6. Implementation Models

In the studies we found in this SR, we noticed that there were several approaches used in designing mobile agent fault tolerance models. These approaches are either integrated into the underlying agent's codes or platforms. Further analysis shows that most of the approaches are a hybrid of a variety of the protocols but some also include preventive techniques. Figure 11 presents the design approaches of existing implementations.

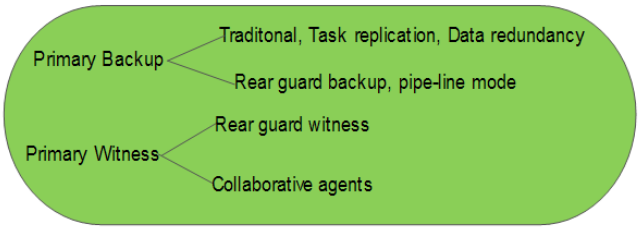
Primary Backup
Traditonal, Task replication, Data redundancy
Rear guard backup, pipe-line mode
Primary Witness
Rear guard witness
Collaborative agents

Figure 11. *Fault tolerance design approach*

4.3.6.1. Primary Back-up Model

The primary back-up model (PBM) design requires the replication of the agents or server into primary component (worker component) and one or more observer components (backups/replicas component). The primary component takes charge of execution while the backup components monitor the primary's computation for any possible failure.

A. Traditional Primary-Backup/Task replication/Data redundancy: In this design, the traditional primary-backup model relies on the replication of system components, task, data, etc. in order to achieve fault tolerance in mobile agents.

B. Rear Guard Agents as Backup: It is chiefly based on the primary-backup principle but instead the backup agent resides on the previously visited host. The backup monitors the primary agent and perform recovery actions to resume the computation when it detects failure of the primary agent especially in a stage-based partition network. We found several forms of these approaches such as multiple rear-guard in [5] and parallel processing of replicas by [44]. Others are optimizations to the approaches by [43] based on rear-guard protocol and reliable broadcast protocol with election protocol, while [7] discusses the pipe-line mode.

4.3.6.2. Primary Witness Model

The primary witness model (PWM) approach is very similar to the PBM, but here the backup agent is a different type of agent usually for monitoring and creating a backup agent when the primary agent fails. The backup agent that replaces the failed primary agent is only created after the detection of a failure. These approaches include rear guard agents as witness and the collaborative agents.

A. Rear Guard Agents as Witness: The witness agent involves a different type of agent that cannot on its own take over as a primary agent during recovery. The primary agent takes charge of the natural execution while the witness only monitors and recovers. In this case, it creates a new primary and restores normal execution. We found this approach in studies such as Monitor Agent (MA) which detects failure of Execution Agent (EA), it creates Repair Agent (RA) for fixing the error in the EA host [31], and the actual and witness agents that creates a probe agent for recovering log during recovery[27], [34].

B. Collaborative Agents: Here, three or more types of agents with designated responsibilities in the detection and recovery processes work together for achieving fault tolerance action with a clear division of labor. The approach involves the primary agent in charge of the execution, while others participate for specifically task such as monitoring, tracking, checking path, recovering, creating another type of agent, etc. Other agents cannot assume the primary agent position during recovery. In this study, studies that used this approach are [25] using three agent types: observer agents, ping agents and transaction agent used for monitoring, path checking and executing transactions respectively. Also, [26] uses three types of agents which are worker, monitor and tracker.

4.4. Mobile Agents Fault Tolerance Performance Factors

SRQ4. What factors influences mobile agent's fault tolerance execution?

We noticed in this study that mobile agent's fault tolerance execution is affected by several factors leading to a decrease in their performance as reported in most studies' experimental data. However, we only focused on the factors rather than the actual figures from the experiments due to unique scenarios and measurements that cannot be quantitatively compared across the studies. From these studies, the influencing factors are as follows:

4.4.1. Agent Size

Agents are affected by the size of their code and the payload data it takes with itself during mobility. In this study, more than 50% of the experimental studies reported agent size as a performance factor and that the execution time of the agents linearly increases with the increase in size of the agent [14], [40], [41], [42], [36], [46], [7]. In addition, increase in size also lead to replication process having overheads [7],[40].

4.4.2. Numbers of Replica

Also more than 50% of the replication based schemes reported that the number of replica affects their performance especially in the synchronous replication schemes [40]. For instance, a design with consensus requires an increase in agent's replicas leading to agents spending longer time than expected. In this SR, the studies that reported number of replica/witness factor in their findings are [40], [41], [36], [46], [39], [28], [34].

4.4.3. Message Size

In this SR, over 30% of the experimental studies consider message size as a factor influencing agent's performance which in turn contributes significantly to network traffic. Increase in the time spent to send a message, increases the size of the messages. We found that, the cost is relatively higher in synchronous communication models than in asynchronous model. Studies that reported message size factor in their findings are [14], [38], [45], [34].

4.4.4. Number of Messages

The number of uncontrolled messages can overwhelm fault detection capability and reducing number of messages can produce performance gain [7]. Some studies that reported the number of messages factor in their findings are [44], [39],[31],[34].

4.4.5. Number of Hops/Host

With agent's characteristics, their survivability decreases with increase in the number of servers the agent's have to visit [34]. This is believed to have impact on reaching a timeout when the execution time takes a long time especially for schemes that relies on timeout and can initiate an unnecessary recovery process that would affect the performance of the system negatively. In this SR, studies that reported this problem in their findings are [7], [26], [27], [44].

4.4.6. Frequency of Check-pointing

Lastly, another important influencing factor is the degree at which check-pointing are taken. An increase in check-pointing frequency increases overhead while infrequent check-pointing brings about much re-computation in the event of recovery [6],[5].

4.5. Platforms Supports

SRQ5. How much supports are offered by the mobile agent's platforms used in implementing the fault tolerance features?

A platform in this context is an executing environment for mobile agents. In this SR, we found varieties of agent's platforms which are mostly built from Java. In addition, most of the existing platforms provide partial or incomplete facilities for fault tolerance mechanisms. We noticed that several academic and commercial systems are available which differs in their features, architecture and implementations, but more or less offer common facilities for the support of agent migration, inter-agent communication, various forms of security and programming or interpreted languages etc.

Analysis here is based on a qualitative comparison of the current agent platforms. Table 4 presents the various platforms used and the nature of fault tolerance support they have in their implementation.

Table 4. *Qualitative comparison of Platforms support for fault tolerance*

Platform	Prog. Lang.	Mobility	Communication	Fault Tolerance Feature
Aglet [58]	Java	Weak	Asynchronous, Synchronous, Proxy	NA
Concordia [59]	Java	Weak	Asynchronous	Yes (Checkpoint, Transactional message queue, Proxy)
FIPA-OS [60]	Java	Weak	Asynchronous	Yes (Replication(clone) Transactional message queue, Proxy)
Grasshopper [61] **	Java	Weak	Synchronous, Asynchronous, Multicast, Dynamic method invocation	NA
JADE [62]	Java	Weak	Asynchronous	NA
JAMES [21]	Java	Weak	JavaSpace	Yes (Checkpoint)
MadKit [63], [64]	Java	Weak	asynchronous message passing	Yes (congestion management, agent monitoring mechanisms)
MOLE [65]	Java	Weak	Asynchronous, Synchronous, Sessions	Yes (Transactional message queue)
Naptel [66]	Java	Weak	asynchronous message passing	Yes (agent monitoring mechanisms, cloning)
Tacoma [68]	C/C++, ML, Perl, python	Weak	Asynchronous, Synchronous	Yes (Rear guards)
Voyager [69]	Java, C#, C++	Weak	Asynchronous, Synchronous, Multicast, Proxy	NA

Note: NA (Not Applicable) implies that either the information or the feature is unavailable. ** Authors have stopped updating the platform

4.6. Challenges

SRQ6. What challenges exist and how do they affects the implementation of the fault tolerance in mobile agents?

There are lots of challenges that faced in existing fault tolerance implementations that has limited their efficiency or the direct application of fault tolerance strategies. Some of the challenges found in this SR are discussed as follows:

4.6.1. Reliable Fault Detection

The key challenge in fault detection is when a fault tolerance process wrongly detects fault and acts upon it. In such cases, an agent is wrongly assumed failed and a replacement agent is created in place of the failed agent leading to the violation of the *"exactly once"* property of agent execution [25]. In this study, we see that existing fault tolerance schemes, rely on techniques such as timeout, periodical exchange of heartbeat message, call backs, etc. for reliable detection of faults but none is absolute in detecting the occurrence of failures. Only 20% of the studies specifically mention this problem and considered it as a very critical attribute.

4.6.2. Network Partition

This is due to communication failure where agent's implementations execute in stages and internal network partitioned into stages or domains. In the event of communication failure, the various stages will be unable to communicate either to advance to the next stage or complete the assigned task, which results in blocking. In this study, only about 12% of the studies specified network partition as a challenge while most studies regard it as a temporary problem.

4.6.3. Lack of Full Process State Capture/Restore

This study found that majority of the existing agent platforms are Java-based systems using the Java Virtual Machine (JVM). With these platforms, analysis indicates that about 90% of them do not allow the capturing and restoring of the full execution state of a process which in turn affects strong mobility support.

4.6.4. Lack of Interoperability

This study noticed fault tolerant agents developed for one agent platform cannot be ported easily to a different agent platform, which impacts the adoption and full realization of fault tolerance. We found few standards such as Foundation for Intelligent Physical Agents (FIPA) and Mobile Agent System Interoperability Facility (MASIF) and steps have been taken to realize this issue in the future.

4.6.5. Lack of Full Transactional Support

This study found that there is no full transactional support in existing fault tolerant framework especially for various types of failure situations. This could be due to the active nature of mobile agent systems which is somewhat incompatible with the concept of transactions that forces several agents to remain in the same transaction as long as it does not commit or abort [6], [29],[69].

4.6.6. Scalability

This study found that all the studies are from the academia and the implemented mobile agent's fault tolerance were not large scale. Analysis shows that most experiments and simulations performed are on a small scale setup, though some promise scalability [26], [33], [41], [44]. In all the studies, we did not find large scale experiments.

4.6.7. Lack of Transparency

In majority of the studies we noticed that developers viewed the fault tolerance platforms as not being transparent since it requires the modification of the underlying platform to accommodate the protocols.

5. Discussion

Despite the importance of mobile agents in distributed environments coupled with the increase in research activities in both academia and industry, we found several issues that still affect the complete realization of reliable fault tolerance in mobile agents. In this study, we noticed that the distribution of the publications by year shows sharp increase between 2003 and 2004 and more than 65% of the studies were conducted between 2004 and 2006. This could be due to the increased interest in the area of mobile agents from the research community. However, the trend seems to be going down resulting in only 23% of the studies being conducted in the last three years, which we think is related to the problem of trust in mobile agents as a result of increasing threat from viruses and network worms. Accordingly, the number of studies that were supported by simulation or experiment improved over the years. They provide detailed description of a model used and reporting some results from a real experiment or simulation to support their findings. Also the existing implementations have classified agent's failures into communication, crash and agent software failure. Each study provides mechanism to detect these failures based on their fault model strategy, but no known fault model can detect and recover from all types of failures in the existing implementations.

It is evident that different approaches exist that deals with the recovering from failure of mobile agents. These approaches include the check-pointing, replication schemes or the hybrid approach and both were used almost equally over the years of consideration. Among the replication-based schemes, most of the studies tend to favor inactive replication due to it lower computation overhead. In the same vein, the mode of agent's communication in existing implementations is mostly asynchronous, as well as there appears to be a balance between transactional or non-transactional execution of mobile agents. The reasons could be from the fact that transactional executions are more reliable in maintaining the exactly-once property, while the non-transactional maintain lower computation overhead. In all the studies we considered, there were other elements such as fault tolerance protocols and their design models. Protocols were based on the exactly-once or non-blocking properties, while the implementation design models were based on hybrid approaches dominated by preventive techniques of agent's replications: primary backup and witness approaches. Each approach has its own strategy to implement fault tolerance.

Other issues we observed were challenges emanating from platform supports and the general challenges in terms of performance affecting mobile agent's fault tolerance that has not been addressed. Variety of mobile agent platforms exist and are dominated by Java which does not support strong mobility of agents which affects full capturing of the state of collaborating processes. Also, the numbers of replicas, messages, size, etc. were among the factors reported as having impacts on the performance of mobile agent's execution.

The consequence of this study is that investigating intensely in the state-of-the-art fault tolerance approaches and the challenges in mobile agent will serve as a starting point and give a clear direction to future researchers who will work in this area to improve the existing implementations.

5.1. Strengths and Weaknesses

In this SR, we have covered several numbers of articles, whose authors are listed in Table 1.We are pretty sure that this study truly covers fault tolerance in mobile agents that have been published to date we considered. We have strictly followed Kitchenham et al. [23] guidelines starting from the planning to the reporting of the review results. However, possible threats or weaknesses in this SR could be related to bias in publication, selection of the included studies and the inaccuracy in data extraction.

For publications, we used sources that are credible and trusted by the research community and also conducted trial searches. We also believe publications from 2009 till date that were not considered will not affect the validity of our study since another study will be performed to compare the trends. In addition, though is possible that some relevant papers may have been missed and if they do, we are sure they are not many and their absence has no significant effect to the information reported in this study. Another issue is the search terms we used. If the search terms/strings formulated were not sufficient and effectively utilized, we believe it has no counter effect on this study. However the two authors worked in collaboration with a librarian and all the selected studies analyzed. In addition, all decisions and results were checked and re-checked.

6. Conclusions

With the increase in system dependability and the exponential growth of the Internet, there is an increasing need to develop reliable mobile agent's fault tolerance techniques capable of withstanding unfavorable and unpredictable behavior of today's systems. Several well known approaches and models for failure detection and recovery schemes have been designed, but none is generic. When deciding on how to develop a reliable fault tolerance in mobile agents, it is important to have the knowledge of the existing techniques, and to be able to make a good decision, taking into account the strong and weak points of diverse approaches against each other. To this effect, this SR identifies the existing fault tolerance mobile agent's approaches and studied them from the perspectives of: establishing facts in the directions of fault tolerance in mobile agents, recognized techniques/approaches, influencing factors, platform supports and challenges. These were chosen because they together helped in giving a good understanding of the existing findings, identify gaps in existing research and provide recommendations for new research activities.

We have analyzed the existing studies implementations in order to realize the state-of-the-art in mobile agent fault tolerance and trends. In this study, we present the found fault tolerance in mobile agents existing implementations in Table 1. In addition, we have developed taxonomy of the existing protocols and the design model used (Figure 10 and 11). Based on the analysis and results obtained, the study takes a closer look at the available mobile agent's fault tolerance strategies and the challenges that affects its realization. The cardinal findings are:

- Agent's fault tolerance fault models generally fall into the three failure types, namely communication, crash and agent software failure though, stated differently in different studies. For instance, place failure, node crash, hardware failure, server failure, host failure; etc were all described as crash failure in the studies.
- Fault tolerance strategies can not address all single-point-of-failures in a system with respect to faults resulting from communication, agent software and crash failures. The complexity and cost of addressing all single-point-of-failures makes the fault tolerance process virtually incomplete. Thus, single-point-of-failures are inevitable in mobile agent and no approach is considered the best in all failure situations.
- Fault tolerance in mobile agent systems can be achieved in a number of ways: replication schemes, check-pointing schemes or a combination of both replication and check-pointing schemes called hybrid approach. However, there is no fault tolerance scheme that is best for all situations since the suitability of an approach heavily depends on the application domain.
- Performance overhead of a fault tolerance strategy is inversely related to recovery time. That is the shorter the recovery time the higher performance overhead. Fault tolerance strategy should try to make a balance between recovery time and performance overhead.
- Mobile agent's fault tolerance faces enormous challenges such as lack of adequate support from agent's platforms as well as lack of resource control capabilities which impacts greatly the realization of a reliable agent execution and need to be drastically addressed.
- Fault tolerant mobile agent is gathering momentum without a clear general framework in its agent system. The existing fault tolerance designs are designed to handle a particular set of fault models and not faults in all situations.

Future work includes proposing a general framework for fault tolerance in mobile agents. This will contribute positively to a high level of system dependability and in addressing the challenges and influencing factors affecting the existing fault tolerance models. In addition, we will validate and implement the framework, perform a more in-depth analysis to investigate challenges outside the framework and investigate fault tolerance in other areas.

Based on the above finding, our recommendations are as follows:

- A generic fault tolerance in mobile agents should be designed and developed since it is difficult to measure the completeness of a fault tolerance approach. The fact is that the existing implementations found in this study focus on partial list of failure types, indicating agents cannot tolerate failure of all types. Thus, issues of all single-point-of-failures should be addressed if high reliability or availability is to be achieved.
- Larger scale tests in real world applications other than simulations should be applied to the available fault tolerance schemes so as to better demonstrate their reliability, capabilities and effectiveness. This is because the maturity of fault tolerance approaches depends highly on the level of acceptance of the approaches. The more the approaches are used, the more the approaches evolve and develop and proved that they work.
- The effects of very long itinerary, many collaborative agents, many replicas, many uncommitted transactions, etc during scalability would need to be investigated so as to increase support for the dependability of fault tolerance techniques.
- Existing platforms should be improved by standardizing and including some of the vital fault tolerance protocols such as cloning and resource monitoring in order to reduce interoperability. Moreover, it will be more flexible to detect runaway agents from within the platform than building a

separate architecture on top of the platforms.

- A better flexibility is if the platforms have a mechanism to selectively apply a fault tolerance protocol to a specific agent or agent place as needed since not all fault tolerance protocols are needed at all times. Developers of agent systems should be able to pick their suitable protocol within their application domain.
- Researchers should focus more on carefully analyzing the unique characteristics of existing agent systems and the applications that are built on them in order to avoid the selection of unsuitable methodology for the recovery of the applications running the agent from faults that affects agents' execution, migration, and interaction.
- Additional fault tolerance features should be introduced for a better reliability.
- Also, alternative agent's design approaches should consider minimizing resource consumption such as using existing code in previously visited hosts instead of re-transporting code improve the reusability of code.
- Lastly, developers should have in mind that the application of fault tolerance approach in the real world applications could also introduce a different set of challenges that have never been thought of.

References

[1] W. Qu and H. Shen, Analysis of Mobile Agents' Fault-Tolerant Behavior, *IEEE/WIC/ACM, Proceedings of International Conference on Intelligent Agent Technology*, 2004, pp 377 – 380, ISBN: 0-7695-2101-0

[2] L. L. Pullum, Software Fault Tolerance Techniques and Implementation, *Artech House*, 2001, ISBN 1-58053-137-7

[3] N. M. Karnik and A. R. Tripathi, Design Issues in Mobile Agent Programming Systems, *IEEE Concurrency*, Vol. 6 , No. 3, 1998, pp 52-61, ISSN:1092-3063

[4] W. Dake and C.P. Leguizamo and K. Mori, Mobile Agent Fault Tolerance in Autonomous Decentralized Database Systems, *IEEE, Proceedings of the Autonomous Decentralized System on The 2nd International Workshop*, 2002, ISBN:0-7803-7624-2

[5] G. Jin, B. Ahn and K. D. Lee, A Fault-Tolerant Protocol for Mobile Agent, *Springer, Proceedings of International Conference on Computational Science and Its Applications*, 2004, pp. 993–1001, ISBN 978-3-540-22057-2

[6] G. Serugendo and A. Romanovsky, Designing Fault-Tolerant Mobile System, *Springer, International Workshop on Scientific Engineering for Distributed Java Applications*, 2003, pp 185-201, ISBN: 978-3-540-00679-4

[7] S. Pleisch, and A. Schiper, Fault-tolerant Mobile Agent Execution, *IEEE, IEEE Transactions on Computers*, Vol. 52 , Nr. 2, 2003, ISSN: 0018-9340

[8] T. Park, I. Byun, H. Kim and H. Y. Yeom, The Performance of Checkpointing and Replication Schemes for Fault Tolerant Mobile Agent Systems, *IEEE Computer Society, Proceedings of 21th IEEE Symposium on Reliable Distributed Systems*, 2002, ISBN:0-7695-1659-9

[9] W. Qu, H. Shen and X. Defago, A Survey of Mobile Agent-Based Fault-Tolerant Technology, *IEEE, Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, 2005, pp 446-450, ISBN:0-7695-2405-2

[10] J. P. Briot, S. Aknine, I. Alvarez and Z. Guessoum, Multi-Agent Systems and Fault-Tolerance: State-of-the-art Elements, *EuroControl Technical Report, LIP6 & MODECO- CReSTIC*, 2007

[11] S. Pleisch, State-of-the-art of Mobile Agent Computing: Security, Fault Tolerance, and Transaction Support, *Research Report, IBM Research, Z. R. Lab. Switzerland*, 1999

[12] S. Pleisch and A. Schiper, FATOMAS-A Fault-Tolerant Mobile Agent System Based on the Agent-Dependent Approach, *IEEE, Proceedings of the 2001 International Conference on Dependable Systems and Networks*, 2001, pp 215-224, ISBN:0-7695-1101-5

[13] L. M. Silva, G. Soares, P. Martins, V. Batista and L. Santos, The Performance of Mobile Agent Platforms, *IEEE, Proceedings of First International Symposium on Third International Symposium on Mobile Agents*, 1999, pp. 270-271, ISBN: 0-7695-0340-3

[14] K. Park, A Fault-Tolerant Mobile Agent Model in Replicated Secure Services, *Springer, Proceedings of International Conference Computational Science and Its Applications*, Vol. 3043, pp 500-509, 2004, ISBN 978-3-540-22054-1

[15] P. Braun, and W. Rossak, Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit, *Morgan Kaufmann*, 2005, ISBN-13: 978-1558608177

[16] R. S. Gray, D. Kotz, G. Cybenko, and D. Rus, Mobile Agents: Motivations and State-of-the-Art Systems, *Dartmouth College, Technical Report: TR2000-365*, 2000

[17] D. Kotz and R. S. Gray, Mobile Agents and the Future of the Internet, *ACM, ACM SIGOPS Operating Systems Review*, Vol. 33, Nr. 3, 1999, pp 7-13, ISSN: 0163-5980

[18] D. B. Lange and M. Oshima, Seven Good Reasons for Mobile Agents, *ACM, Communications of the ACM*, Vol. 42, Nr. 3, 1999, pp 88-89, ISSN:0001-0782

[19] M. Eid, H. Artail, A. Kayssi, and A. Chehab. Trends in Mobile Agent Applications, *Journal of Research and Practice in Information Technology*, Vol. 37, No. 4, November 2005

[20] R. Boutaba and J. Xiao, Network Management: State of the Art, *Kluwer, B.V., Proceedings of the IFIP 17th World Computer Congress - TC6 Stream on Communication Systems: The State of the Art*, 2002, pp 127-146, ISBN: 1-4020-7168-X

[21] L. M. Silva, P. Simões, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida and N. Stohr, JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks, *Springer, Proceedings of Third International Workshop Intelligent Agents for Telecommunication Applications*, Vol. 1699, 1999, pp 76-95, ISBN 978-3-540-665397

[22] O. Kachirski and R. Guha, Intrusion Detection Using Mobile

Agents in Wireless Ad Hoc Networks, *IEEE, Proceedings of the IEEE Workshop on Knowledge Media Networking*, 2002, pp 153-158, ISBN:0-7695-1778-1

[23] B. Kitchenham and S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, *Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001,* 2007

[24] J. W. Creswell and Dana L. Miller, Determining Validity in Qualitative Inquiry, *Theory Into Practice,* 1543-0421, Volume 39, Issue 3, 2000, Pages 124 – 130

[25] S. Summiya, K. Ijaz, U. Manzoor and A. A. Shahid, A Fault Tolerant Infrastructure for Mobile Agents, *IEEE, Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, 2006, pp 235-235, ISBN:0-7695-2731-0

[26] K. K. Leung and K. W. Ng, A fault-tolerance mechanism for mobile agent systems, *Proceedings. 2006 International Conference on Intelligence For Modelling, Control and Automation. Jointly with International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 2005, pp.7 ISBN-10: 0 7695 2504 0

[27] M.R. Lyu and T. Y. Wong, A progressive fault tolerant mechanism in mobile agent systems *SCI 2003. 7th World Multiconference on Systemics, Cybernetics and Informatics Proceedings*, 2003, pp. 299-306 , Vol.9 ISBN-10: 980 6560 01 9

[28] S. Mellouli, A reorganization strategy to build fault-tolerant multi-agent systems *Advances in Artificial Intelligence. 20th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2007. Proceedings 2007, Vol.4509*, pp. 61-72, ISBN-10: 3 540 72664 0

[29] F. M. A. Silva and R. Popescu-Zeletin, An Approach for Providing Mobile Agent Fault Tolerance, *Springer, Proceedings of Second International Workshop on Mobile Agents*, Vol. 1477, 1998, pp 14-25, ISBN: 978-3-540-64959-5

[30] T. Osman, W. Wagealla and A. Bargiela, An Approach to Rollback Recovery of Collaborating Mobile Agents, *IEEE, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 34, Nr. 1, 2004, pp 48-57, ISSN: 1094-6977

[31] X. Meng and H. Zhang, An efficient fault-tolerant scheme for mobile agent execution *First International Symposium on Systems and Control in Aerospace and Astronautics*, 2006, pp.5, ISBN-10: 0 7803 9395 3

[32] J. Chen; H. Shi; C. Chen; Z. Hong; P. Zhong, An efficient forward and backward fault-tolerant mobile agent system, *Eighth International Conference on Intelligent Systems Design and Applications*, 2008, pp.61-6, ISBN-13: 978-0-7695-3382-7

[33] M. Olivier, B. Marin , S. Pierre, G. Zahia and B. Jean-Pierre, DARX - A Self-healing Framework for Agents, *Springer, Third International Conference,* 2007, Vol. 4322/2007, pp. 88-105, ISBN: 978-3-540-71155-1

[34] M.R. Lyu, X. Chen and T. Y. Wong, Design and Evaluation of a Fault-Tolerant Mobile-Agent System, *IEEE, Intelligent Systems*, Vol. 19 , Nr. 5, 2004, pp 32-38, ISSN: 1541-1672

[35] Y. Tanaka, N. Hayashibara, T. Enokido and M. Takizawa, Fault-tolerant distributed systems in a mobile agent model, *Seventeenth International Conference on Database and Expert Systems Applications*, 2006, pp.5, 2006 ISBN-10: 0 7695 2641 1

[36] L. M. Silva, V. Batista and J. G. Silva, Fault-Tolerant Execution of Mobile Agents, *IEEE, Proceedings International Conference on Dependable Systems and Networks*, 2000, pp 135-143, ISBN: 0-7695-0707-7

[37] FIPA, http://www.fipa.org, [Accessed August 26, 2009]

[38] K. Rothermel and M. Strasser, A fault-tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents, *Proceedings Seventeenth IEEE Symposium on Reliable Distributed Systems 1998, pp.* 100-8, ISBN-10: 0 8186 9218 9

[39] A. Fedoruk and R. Deters, Improving Fault-Tolerance by Replicating Agents, *ACM, Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, 2002, pp 737-744, ISBN:1-58113-480-0

[40] T. Park, I. Byun and H. Y. Yeom, Lazy Agent Replication and Asynchronous Consensus for the Fault-Tolerant Mobile Agent System, *Springer, Proceedings of Third International IFIP-TC6 Networking Conference*, 2004, pp 1060-1071, ISBN: 978-3-540-21959-0

[41] T. Park and I. Byun, Low Overhead Agent Replication for the Reliable Mobile Agent System, *Springer, Proceedings of 9th International Euro-Par Conference*, Vol. 2790, 2003, pp 1170-1179, ISBN 978-3-540-40788-1

[42] H. Hamidi and K. Mohammadi, Modeling Fault Tolerant and Secure Mobile Agent Execution in Distributed Systems, *Idea Group, International Journal of Intelligent Information Technologies*, Vol. 2, Nr. 1, 2006

[43] D. Johansen, K. Marzullo, F.B. Schneider, K. Jacobsen, and D. Zagorodnov, NAP: practical fault-tolerance for itinerant computations, *Proceedings. 19th IEEE International Conference on Distributed Computing Systems 1999, pp.* 180-9, ISBN-10: 0 7695 0222 9

[44] J. Yang, J. Cao, W. Wu and C. Xu, Parallel algorithms for fault-tolerant mobile agent execution, *Distributed and Parallel Computing. 6th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP. Proceedings,* 2005, pp. 246-56, ISBN-10: 3 540 29235 7

[45] M. Tosic and A. Zaslavsky, Reliable multi-agent systems with persistent publish/subscribe messaging, *18th Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2005. Proceedings, 2005 Vol. 3533*, pp.165-74, ISBN-10: 3 540 26551-1

[46] P. Taesoon and Y. Jaehwan, The K-Fault-Tolerant Checkpointing Scheme for the Reliable Mobile Agent System, *Parallel and Distributed Computing: Applications and Technologies. 5th International Conference, PDCAT 2004. pp.*577-81, ISBN-10: 3 540 24013 6

[47] T. Kaneda, Y. Tanaka, T. Enokido and M. Takizawa, Transactional agent model for fault-tolerant object systems, *Proceedings of the ACM Symposium on Applied Computing*, 2005, Vol. 2, pp. 1133-1138

[48] T. Park, J. Youn and D. Kim, Using adaptive agents for the fault-tolerant mobile computing system, *6th International Conference, Proceedings* 2006, Vol. 3993, pp. 807-814 ISBN-10: 3540343830

[49] H. K. Yeom, H. Y. T. Park and H. Park, The Cost of Checkpointing, Logging and Recovery for the Mobile Agent Systems, *Proceedings of Pacific Rim International Symposium on Dependable Computing*, 2002, pp 45-48, ISBN: 0-7695-1852-4

[50] J. Briot, Z. Guessoum, S. Charpentier, S. Aknine, O. Marin and P. Sens, Dynamic Adaptation of Replication Strategies for Reliable Agents, *Proceeding of 2nd Symposium on Adaptive Agents and Multi-Agent Systems*, 2002, pp 10-19

[51] M. Q. Patton, Qualitative Research & Evaluation Methods, *Sage Publications, Inc; 3rd edition*, 2001, ISBN-10: 0761919716

[52] E. N. (Mootaz) Elnozahy, L. Alvisi, Y. Wang and D. B. Johnson, A Survey of Rollback-Recovery Protocols in Message-Passing Systems, *ACM, ACM Computing Surveys*, Vol. 34, Nr. 3, 2002, pp 375-408, ISSN:0360-0300

[53] S. Mishra and P. Xie, Interagent Communication and Synchronization Support in the DaAgent Mobile Agent-Based Computing System, *IEEE, IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, Nr. 3, 2003, ISSN:1045-9219

[54] L. Bettini and R. D. Nicola, Translating Strong Mobility into Weak Mobility, *Springer, Proceedings of the 5th International Conference on Mobile Agents*, Vol. 2240, 2001, pp 182-197, ISBN:3-540-42952-2

[55] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill and R. Jeffers, Strong Mobility and Fine-Grained Resource Control in NOMADS, *Springer, Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents*, 2000, pp 2-15, ISBN:3-540-41052-X

[56] R. A. Bourne, A. L. G. Hayzelden, Rachel Bourne and P. Buckle, Agent Technology for Communication Infrastructures, *Wiley*, 2001, ISBN: 0471498157

[57] Aglet, http://aglets.sourceforge.net/, [Accessed May 21, 2009]

[58] T. Walsh, N. Paciorek and D. Wong, Security and Reliability in Concordia, *IEEE, Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences*, Vol. 7, 1998, pp 44-53, ISBN: 0-8186-8255-8

[59] FIPA-OS Tutorial, http://fipa-os.sourceforge.net/tutorials.htm, [Accessed August 26, 2009]

[60] Agents Technology in Europe, *ACTS project InfoWin*, 1999

[61] JADE - Java Agent DEvelopment Framework, http://jade.tilab.com/, [Accessed August 26, 2009]

[62] MadKit, http://www.madkit.org, [Accessed May 21, 2009]

[63] MadKit: A generic Multi-agent platform, *ACM*, 2000.

[64] J. Baumann, F. Hohl, K. Rothermel, M. Strasser and W. Theilmann, *MOLE: A Mobile Agent System*, John Wiley & Sons, Software—Practice & Experience, Vol. 32, 2002, pp 575–603

[65] Naptel, http://www.ece.eng.wayne.edu/~czxu/software/naplet.html, [Accessed August 26, 2009]

[66] A. Grimstrup, R. Gray, D. Kotz, M. Breedy, M. Carvalho, T. Cowin, D. Chacón, J. Barton, C. Garrett and M. Hofmann, Toward Interoperability of Mobile-Agent Systems, *Springer, Proceedings of 6th International Conference on Mobile Agents*, 2002, pp 106-120, ISBN: 978-3-540-00085-3

[67] D. Johansen, R. V. Renesse, F. B. Schneider, N. P. Sudmann and K. Jacobsen, A Tacoma Retrospective, *John Wiley & Sons, Software—Practice & Experience*, Vol. 32 , Nr. 6, 2002, pp 605-619, ISSN:0038-0644

[68] Voyager 3.1.1 Developer Guide, *Object Space*, 1999

[69] M. J. Fischer, N. A. Lynch and M. S. Paterson, Impossibility of Distributed Consensus With One Faulty Process, *ACM, Journal of the ACM*, Vol. 32, Nr. 2, 1985, pp 374-382, ISSN: 0004-5411