Integrated Replication-Checkpoint Fault Tolerance Approach of Mobile Agents "IRCFT"

Suzanne Sweiti[1] and Amal Al Dweik[2]
[1]Deanship of Graduate Studies and Scientific Research, Palestine Polytechnic University, Palestine
[2]College of Information Technology and Computer Engineering,
Palestine Polytechnic University, Palestine

Abstract: *Mobile agents offer flexibility which is evident in distributed computing environments. However, agent systems are subject to failures that result from bad communication, breakdown of agent server, security attacks, lack of system resources, congestion in network, and situations of deadlock. If any of such things happen, mobile agents suffer loss or damage totally or partially while execution is being carried out. Reliability must be addressed by the mobile agent technology paradigm. This paper introduces a novel fault tolerance approach "IRCFT" to detect agent failures as well as to recover services in mobile agent systems. Our approach makes use of checkpointing and replication where different agents cooperate to detect agent failures. We described the design of our approach, and different failure scenarios and their corresponding recovery procedures are discussed. The proposed system is implemented over Agelt platform. The system improves the performance significantly.*

Keywords: *Mobile agents, fault tolerance, reliability, checkpointing, replication*.

Received September 15, 2015; accepted October 18, 2015; published on line January 28, 2016

1. Introduction

In both the academic and industrial trends, mobile agents are very important in the recent trends of distributed computing. They have their own properties which make them flexible in the deployment. As a result, the design, application and maintenance of the distributed systems become a simple task [10, 11].

Like any other software systems, mobile agents are not isolated from operating in unusual situations. They are somewhat subject to fault as they consist of autonomous components in distributed dynamic environments [2]. Mobile agents might come across usual errors which emerge especially during migration request failure, security penetration or communication exceptions [13]. The issue of reliability is extremely important in order to challenge such failures. The goal is to allow the system to work flexibly in spite of the faults which continue to exist in the system after development [10].

In recent years, Multi-Agent Systems (MAS) had started gaining widespread acceptance in the field of information technology [3]. The most important requirements among the approaches to handle the fault tolerance in MAS are: Non-blocking and exactly-once [1]. The non-blocking property guarantees that the agent continues executing to achieve its goal even in case of an infrastructure component. The use of replication may cause the violating of the dominant property of the mobile agent: The exactly once execution of the mobile agent [1]. The objective of the protocols for dealing with the exactly-once property is to ensure that the agent has to carry out the intended action one time only in a host.

The remainder of this paper is organized as follows: The next section introduces the previous work. Section 3 defines our model, different failures and recovery scenarios are discussed in section 4. The characteristic of the presented approach are discussed in section 5. In section 6, a comparison between approaches is done. Implementation is introduced in section 7. Finally, the conclusion is introduced in section 8 followed by the references.

2. Related Works

Singh and Dave [14] have proposed an approach for providing efficient fault tolerance in mobile agent systems to overcome certain failures. The parallel checkpointing approach is used which considers the antecedence graphs. The used graphs are directed acyclic graphs that record the dependency information.

Hans and Kaur [5] have proposed an approach to sort out the agent crash problem. The clone of original agent is used in an itinerary to follow the actual agent. So, if any failure occurs in the mobile agent system, the recovery is possible by the clone.

The main aim is to limit the rollback by adding checkpoints. Rostami *et al*. [12] suggested that the server and agent failures are detected and recovered by the cooperation of agents with each other. In order to detect and recover the failed agent in 2-Dimensional

Mesh Network, another types of agent are used, namely the witness agent, to monitor whether the actual agent is alive or dead. However, the idea of agent tracking technique is described in [9]. It provides an efficient system where excellent management and maintenance can be performed in the networks of mobile agents.

Zeghache and Badache [15], the fault tolerant mechanism has been laid with relation to the applications that deal with transactions. The protocol proposed is based on the behaviour of mobile agent, Watch Agent as well as Transaction Manager. The designing of adaptive mobile agents [8] aims at accepting additional roles when working inside a special environment that is called context-aware environment.

Kaur *et al.* [6], an integrated mechanism has been proposed using Secure Mobile Agent Platform System (SMAPS) which aims to prevent agent blocking in certain cases where agent is identified by malicious host. Then, it is used to follow the location of the mobile agents during the process at any time.

Hans and Kaur [4], the agent put its computation results on the home server after completing its task on first three servers in its itinerary. The approach makes use of check pointing, partial results and the address of last host visited is saved prior before the agent visits the next host in the itinerary.

Lyu and Wong [7] raised another fault-tolerant model based on Witness. This model employs three types of agents. Discovery of failure is done by witness agent and recovery is message log based recovery and also checkpointing.

3. Integrated Replication-Checkpoint Fault Tolerance Approach (IRCFT)

IRCFT approach is a development of that implemented in [4, 7]. We address a fault tolerance approach of deploying cooperating agents. The basic idea used in the work is to tolerate faults using the concept of checkpoint and replication.

3.1. Assumptions

The agents in IRCFT communicate at different locations by exchanging messages through unreliable communication channels. Therefore, the system is assumed to use unreliable network connection. Some specific assumptions in the system may be summarized in the following points:

1. Agents in the system can be generated from every server on network.
2. The home server is always available and free of failure.
3. No failure in log entries and the mobile agent can be recorded in the permanent storage.
4. No Byzantine failure.
5. When a server failure occurs in S_i, all the agents inside S_i will be terminated.

3.2. IRCFT Description

In our approach, we distinguish four types of agents; one type is performing the required computation for the user, named as Worker Agent (WA). Another type is to detect the status of the actual agent, named as Monitor Agent (MA). It is responsible to monitor the status of other monitors. The third agent is the Manager agent which responsible to send the address of the next server to WA. The last agent's is the replica agent to recover WA. MA and WA communicate by using a peer-to-peer messages passing mechanism. MA sets a timer with a certain time-out value for each server S_i. We also need to log the actions performed by the WA. The information logged by the agent is vital for failure detection.

During an agent's trip, the agent will visit a number of servers. The sequence of servers visited composes the agent's itinerary. The servers on the itinerary are denoted by S_0, S_1, S_2…, S_n, where S_i is the i^{th} visited server. Specially, S_0 is the home server on which the agent is generated. At first, the Manager agent creates WA and MA and they migrate to the next server. Figure 1 illustrates the workflow of the protocol.

Assume that, currently there are *n* network servers on the execution itinerary, and the WA has just arrived at server S_{i+1} and does not reach the fourth server yet, WA and MA are at the third server and the elder MA reside in the server S_i.

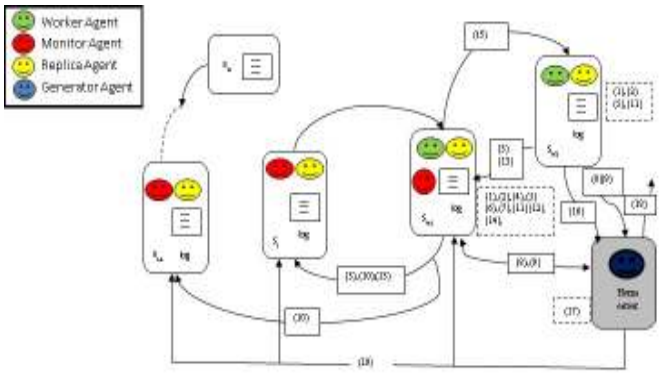

Figure 1. Workflow of IRCFT protocol.

The workflow is as follows:

1. After WA has arrived at S_{i+1}, it immediately registers a log_{arrive} on the permanent storage in S_{i+1}..
2. After WA has arrived at S_{i+1}, it immediately registers a log_{arrive} on the permanent storage in S_{i+1}..
3. MA_{i+1} informs MA_i that WA_{i+1} has arrived at S_{i+1} safely by sending a msg_{arrive} message to MA_i.
4. W_{Ai+1} creates a replica Rc_{Ai+1}. This replica is used when the agent die.
5. Next, WA_{i+1} accomplishes the task appointed by the owner on S_{i+1}.
6. After WA_{i+1} has finished the task, it takes a checkpoint in timeout period (T_{cp}) sends it to

replica. RcA_{i+1} update its status using the checkpoint.
7. WA_{i+1} sends message $msg_{checkpoint}$ to MA_{i+1}. The reason of this message is to inform the MA_{i+1} that RcA_{i+1} is updated its status.
8. WA_{i+1} sends its current address to the Manager agent and waits for a response for address of the next server. If not response it resends the request message.
9. Manager agent sends address of next server if the current address is not in the list of the precedent address visited.
10. RcA_{i+1} sends an update message msg_{update} to the previous replicas; RcA_i and RcA_{i-1}; and waits for reception of acknowledge message (msg_{ack}) from the other replicas to be sure that all replicas are updated.
11. WA_{i+1} registers a log_{leave} in S_{i+1}. This log entry expresses that WA has completes its computation and is ready to travel to the next server S_{i+2}.
12. In the next step, WA_{i+1} sends to MA_{i+1} a message, $msg_{LogLeave}$, in order to inform MA_{i+1} that WA_{i+1} is ready to leave S_{i+1}.
13. MA_{i+1} sends message msg_{leave} to MA_i.
14. When MA_{i+1} receive $msg_{LogArrival}$ and $msg_{LogLeave}$ from WA_{i+1}, it spawns a new monitor in S_{i+1}.
15. WA_{i+1} and the new monitor leave S_{i+1}, and migrate to S_{i+2}. Note that the new witness agent knows where to go, i.e. S_{i+2}, because $msg_{LogLeave}$ contain information about the location of S_{i+2}.

After completing its execution on the first three servers of the itinerary the WA moves to S_{i+2} and repeats the following steps 1-3, 5, 8-9 and 11-13 motioned above. When WA_{i+2} sends the $log_{LogLeave}$ to MA_{i+2}, then step 16 is done.

16. WA_{i+2} moves back to the *home server* and checkpoint the data and saves the values computed from the previous four servers.
17. After saving the value and adding checkpoints, the Manager agent creates a new monitor.
18. The WA and the new monitor move to the next server in the itinerary that is S_{i+3}.
19. Finally, the Manager agent sends msg_{kill} to the RcAs and MAs residing in the previous servers since they are no longer needed.

The process is repeated for every four servers in the itinerary until WA reaches the last destination in its itinerary and it returns back to the home server.

4. Failure and Recovery Scenarios

If the current server is S_{i+1}, Figure 2 shows the different scenarios of failure in the first three servers.

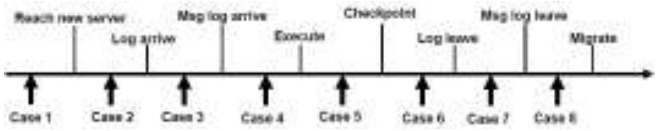

Figure 2. Different scenarios of failure in the first three servers.

4.1. Safe Case

In the safe case, MA_i successfully receive msg_{arrive} and msg_{leave} from MA_{i+1} and the WA_{i+1} successfully completes execution on S_{i+1} and moves safely to the next server S_{i+2}.

4.2. The Monitor MA_{i+1} Fails to Receive $msg_{logarrival}$

- **Case 1, 2, 3 and 8**. The reason can be:
  1. WA_{i+1} is terminated when it is ready to leave S_i.
  2. WA_{i+1} is terminated when it has just arrived at S_{i+1}; without logging; or.
  3. WA_{i+1} is terminated when it has just arrived at S_{i+1} with logging and before sending $msg_{LogArrival}$ to MA_{i+1}.

In this case MA_{i+1} waits for the message $msg_{LogArrival}$ for timeout period ($T_{LogArrival}$). If the timeout is reached, it sends to MA_i failure message $msg_{failure}$. Then, MA_i sends $msg_{failure}$ to RcA_i. RcA_i travel to S_{i+1} to recover the failure. After the recovery is completed, the recovered WA can start performing it computation.

4.3. The Monitor MA_{i+1} Fails to Receive $msg_{checkpoint}$

- **Case 4, 5 and 6**. The reason can be:
  1. WA_{i+1} is terminated when MA_{i+1} has just sends msg_{arrive} to MA_i.
  2. WA_{i+1} is finished its execution; or
  3. WA_{i+1} is finished the checkpoint to update RcA_{i+1}.

In this case MA_{i+1} waits for the $msg_{checkpoint}$ for timeout period ($T_{checkpoint}$). If the timeout period is reached, it sends to M_i a failure message $msg_{failure}$. Then MA_i send $msg_{failure}$ to RcA_i and travel to S_{i+1} to recover the failure.

4.4. The Monitor MA_{i+1} Fails to Receive $msg_{logleave}$

- **Case 7**. The reason can be:
  1. WA_{i+1} is terminated when it has just sends $msg_{checkpoint}$ to MA_{i+1}.
  2. WA_{i+1} is terminated when it has just logged the leave entry.

In this case MA_{i+1} waits for the $msg_{LogLeave}$ for timeout period ($T_{LogLeave}$). If the timeout is reached it sends to MA_i error message msg_{error} to inform MA_i that recovering will be through the replica locates in the server S_{i+1}. The error message will be sent also to RcA_{i+1}. Then, RcA_{i+1} recover the failure.

4.5. The Monitor MA_i Fails to Receive msg_{arrive} or msg_{leave}

1. msg_{arrive} or msg_{leave} is lost due to an unreliable network.

2. msg_{arrive} or msg_{leave} is arrived after the timeout period of MA_i; or.
3. WA_{i+1} and MA_{i+1} are terminated due a crash failure.

If the failure is because of the first two reasons, the WA_{i+1} is still alive in S_{i+1}. In this case MA_i waits for the message for timeout period (T_{arrive} or T_{leave}). If the timeout is reached MA_i sends $msg\_failure_{arrive}$ to RcA_i to verify the log_{arrive}; or $msg\_failure_{leave}$ to verify log_{leave}. Then, RcA_i travels to S_{i+1} to search for log_{arrive} or log_{leave} in S_{i+1}. If found, then RcA_{i+1} re-transmits msg_{arrive} or msg_{leave} to MA_i, then RcA_{i+1} sends a message to MA_{i+1} to verify if it is lost or no. If RcA_{i+1} fails to receive the response, it creates a new monitor and returns to S_i.

If MA_i fails to receive msg_{arrive} because of the loss of MA_{i+1} and WA_{i+1}, it sends $msg\_failure_{arrive}$ to the RcA_i and travel to S_{i+1} to search for log_{arrive}. Upon arriving at S_{i+1}, it searches the log file for the entry log_{arrive}. If the log entry is not found, RcA_{i+1} sends a message to MA_{i+1} to verify if it is died or no. If RcA_{i+1} fails to receive the response, it creates a new monitor, recover the WA_{i+1} and re-transmits the message msg_{arrive} to MA_i.

When MA_i fails to receive msg_{leave} because of the loss of MA_{i+1} and WA_{i+1}, it sends $msg\_failure_{leave}$ to the RcA_i and travel to S_{i+1} to search for log_{leave}. Once the RcA_i reached S_{i+1}, it searches the log file for the entry log_{leave}. If it is not found, RcA_{i+1} sends a message to MA_{i+1} and waits for a response. If it fails to receive the response, it creates a new monitor and recover the worker agent. If the replica receives the update before the failure then it continues the execution, otherwise it repeats the task.

4.6. The WA_{i+1} and RcA_i Fail

In this case MA_{i+1} sends $msg_{failure}$ to MA_i. MA_i detects that RcA_i is terminated then, it re-sends the failure message to MA_{i-1} in the server S_{i-1}. Then, RcA_{i-1} can migrate and recover the failure in S_{i+1}.

4.7. Monitor Failure

In the life cycle of a monitor, the monitor periodically sends a heartbeat message to the last monitor. It is performed considering the following chain:

$MA_{i-2} \rightarrow MA_{i-1} \rightarrow MA_i \rightarrow MA_{i+1} \rightarrow WA_{i+1}$.

If MA_{i-2} does not receive the heartbeat message; $msg_{heartbeat}$ from MA_{i-1} after several attempts, then it suspects that MA_{i-1} is down. The monitor MA_{i-2} will request the identity and the server address of the next monitor from the Manager Agent. Once MA_{i-2} receive the identity and the address of the next server, it will exclude MA_{i-1} from the chain by linking with monitor MA_i.

If MA_i receives $msg_{NewLink}$ from a MA_{i-2}, it will consider it as the last monitor and reply with $msg_{heartbeat}$. If MA_{i-1} does not receive $msg_{heartbeat}$ from MA_i. Then it suspects that MA_i is down. In this case, MA_{i-1} will request the identity and the server address of the next monitor from the Manager agent. The Manager agent will send the next address without the identifier. Then, MA_{i-1} send $msg_{NewLinkWorker}$ and only MA_{i+1} receive the message and reply with msg_{arrive} or msg_{leave}.

4.8. Bad Servers

When bad servers increase in the itinerary, both of WA, and the monitors and replicas are down. In this situation, the checkpoint in the home server recovers the failure. After a period of time, a fault message is sent to Manager Agent. It creates a new monitor and a replicated copy of the agent to starts its execution from the immediate checkpoint saved in the home server.

4.9. In the Fourth Server

When the WA migrates to the fourth server and terminates the execution, the RcAs and the MAs in the previous servers will be killed, then if the WA stops its execution due to any fault when it is in the next server. In this situation, a fault message is sent from the MA to Manager agent in the home server and we have no other option than to send the replicated copy of the agent. The data retrieved from the previous server is already saved to the home server with the checkpoints after every four servers. So, the replicated agent needs not to rollback to the first server in the itinerary. The replicated starts its execution from the immediate checkpoint.

5. Discussion

5.1. Preservation of the Exactly-Once Property

The exactly-once execution property is guaranteed in IRCFT. The agent replicas are not executing while the original executing agent is active, and it is impossible for more than one WA to gain the acceptance from the Manager agent for the same stage. The duplicated agent will be terminated.

5.2. Preservation of Non-Blocking Property

The non-blocking feature is guaranteed even in the case of multiple failures by allowing the last checkpoint or the replica of the crashed agent to replace it in order to continue execution even in the case of agent failures.

5.3. Management of Monitors Failure

In this approach, new link is used to maintain the chains of monitors. The servers with high crash rate are eliminated from the chain; because IRCFT approach update the replica each time after the completion of task to be used to recover the failure.

6. Main Differences between IRCFT and Other Approaches

IRCFT approach is a development of that in [4, 7]. IRCFT approach differs from these approaches in several points that are summarized as follows:

1. IRCFT and the approach in [7] communicate at different locations by exchanging messages through unreliable communication channels but in the approach [4] the authors use the reliable network.
2. IRCFT and the approach in [4] use the checkpoint in the home server; checkpoint serves to save partial results after completing the execution in a number of servers of the itinerary. In IRCFT, the checkpoint usage was after completing its execution on the first four servers, but in [4] use it after three servers.
3. In [4, 7] the mobile agent knows its itinerary. In IRCFT the worker agent requests the address of next server from the Manager agent. For this reason, the exactly-once feature is guaranteed. In [4] the exactly-once property is violated due to network congestion.
4. IRCFT minimizes the monitor agents and replicas by terminating them by the Manager agent after the checkpoint in the home server; because existence of all of MA is not necessary on the initial servers [7].
5. Lyu and Wong [7] recovers the lost monitor. This is achieved by preserving the monitoring dependency: the recovery of MA_{i-1} can be performed by MA_{i-2}. MA_{i-1} will be created in order to replace the lost MA in the server S_{i-1}. In IRCFT recovery of monitor is ignored.
6. Lyu and Wong [7] stores the checkpoint data in a stable storage, it is used in case of failure. IRCFT uses the checkpoint to update the replica agent located in the server and the other previous replicas.
7. In case of failure in [4], the checkpoint in the home server is used to recover the failure and the replicated copy of the original agent will be sent to the immediate checkpoint before the fault. But in IRCFT, the update replica is used to recover the failure, and it uses the checkpoint in the home server to recover the failure in case where all replicas are terminated and there is no way to recover the failure using the update replica.

7. Implementation

IRCFT using AGLETS-2.0.2 where we evaluated the round trip time (time required by mobile agent to complete its itinerary) and the agent survivability. The system is run for 30 times for each experiment, then, the average then was taken for each experiment. The results are shown in Tables 1, 2, 3, and 4.

Table 1. Effect of round trip time when there is no failure.

Server	4	7	9	13	16	20
IRCFT	7662	12838	16429	23554	29037	36162

Table 2. Effect of round trip time by considering several agent failure.

Failure in the Server	3	3, 7	3 , 7, 11	3, 7, 11, 15	3, 7, 11, 15, 20
IRCFT before Checkpoint	6768	15265	23762	32259	42725
IRCFT after Checkpoint	5591	12911	20231	27551	38017

Table 3. Effect round trip time by considering several agent failure and blocking of the previous server.

Failure in the Server	3	3, 7	3 , 7, 11	3, 7, 11, 15	3, 7, 11, 15, 20
IRCFT before Checkpoint	6918	15565	24212	32859	43288
IRCFT after Checkpoint	5591	12911	20231	27551	38017

Table 4. Survivability.

N° of Servers	1	4	8	12	16	20
Survivability	1	1	0,96	0,93	0,9	0,87

8. Conclusions

The IRCFT approach shows that it can improve and enhance the survivability of the agent and it will diminish the time needed for detecting faults and repairing failure. Then the transmission of mobile agent to the next server will be more reliable. Furthermore, it will decrease trip time when errors occur.

The agent replicas and checkpoint are not executing while the original executing agent is active. Therefore, only one execution of the agent will be guaranteed at the same time. This property ensures the exactly once execution which is the most important feature for the agent execution. The non-blocking feature is also guaranteed even in the case of multiple failures by allowing the last checkpoint or the replica of the crashed agent to replace it in order to continue execution even in the case of agent failures. Hence, both checkpoint and replication is introduced in the suggested approaches to solve the blocking problem of the mobile agent execution where the replication and checkpoint masks failures and ensures progress of the mobile agent execution.

For future work, the approaches suggested in [4, 7] will be implemented and compared to our developed approach.

References

[1] Bellifemine F., Caire G., and Greenwood D., *Developing Multi-Agent Systems with JADE*, John Wiley and Sons Ltd, England, 2007.

[2] Dake W., Leguizamo C., and Mori K., "Mobile Agent Fault Tolerance in Autonomous Decentralized Database Systems," *in Proceeding of IEEE Autonomous Decentralized System on The 2nd International Workshop*, Washington DC, USA, 2002.

[3] Elamary M. and Issa Z., "Using Model Driven Architecture to Develop Multi- Agent Systems,

the International Arab Journal of Information, vol. 10, no. 4, pp. 349-355, 2013.

[4] Hans R. and Kaur R., "Fault Tolerance Approach in Mobile Agents for Information Retrieval Applications Using Check Points," *International Journal of Computer Science and Communication Networks*, vol. 2, no. 3, pp. 347-353, 2012.

[5] Hans R. and Kaur R., "Novel Dynamic Shadow Approach for Fault Tolerance in Mobile Agent Systems," *in Proceeding of 6th International Conference on Signal Processing Communication Systems, Publication IEEE Conference*, Gold Coast, Australia, 2012.

[6] Kaur R., Challa, R., and Singh, R., "Integrated mechanism to Prevent Agent Blocking in Secure Mobile Agent Platform System," i*n Proceeding of International Conference on Advances in Computer Engineering*, Karnataka, India, pp. 158-162, 2010.

[7] Lyu M. and Wong T., "A Progressive Fault Tolerant Mechanism in Mobile Agent Systems," i*n Proceeding of 7th World Multiconference on Systemics, Cybernetics and Informatics*, vol. 9, pp. 299–306, 2003.

[8] Marikkannu P., Adri-Jovin J., and Purusothaman T., "Fault-Tolerant Adaptive Mobile Agent System using Dynamic Role based Access Control," *International Journal of Computer Applications*, vol. 20, no.2, 2011.

[9] Mitrović D., Budimac Z., Ivanović M., Vidakovic M., "Improving Fault Tolerance of Distributed Multi-Agent Systems with Mobile Network-Management Agents," i*n Proceeding of the International Multiconference on Computer Science and Information Technology*, Wisla, Poland, pp. 217-222, 2010.

[10] Pullum L., *Software Fault Tolerance Techniques and Implementation*, Artech House, USA, 2001.

[11] Qu W. and Shen H., "Analysis of Mobile Agents, Fault-Tolerant Behavior," *in Proceedings of IEEE/ WIC/ ACM International Conference on Intelligent Agent Technology*, pp. 377-380 ,2004.

[12] Rostami A., Rashidi H., and Zahraie M. , "Fault Tolerance Mobile Agent System Using Witness Agent in 2-Dimensional Mesh Network," *International Journal of Computer Science* Issues, vol. 7, no. 5, pp. 153-158, 2010.

[13] Serugendo G. and Romanovsky A., "Designing Fault-Tolerant Mobile System," *Springer, International Workshop on Scientific Engineering for Distributed Java Applications*, London, UK, 2003.

[14] Singh R. and Dave M., "Antecedence Graph Approach to Checkpointing for Fault Tolerance in Mobile Agent Systems", *IEEE Transactions on Computers*, vol. 62, no. 2, 2013.

[15] Zeghache L. and Badache N., "Optimistic Replication Approach for Transactional Mobile Agent Fault Tolerance," i*n Proceeding of 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing*, London, UK, pp. 193-198, 2010.

Suzanne Sweiti received her MSc degree of Informatics in 2015 from Palestine Polytechnic University, Hebron, Palestine, and BCs degree in Computer Engineering from University Abderrahmane Mira Bejaia, Algeria in 2008.

Amal Al Dweik received her PhD of Information Networks from Cairo University. Currently, she is working as assistant professor in College of Information Technology and Computer Engineering in Palestine Polytechnic University, Hebron, Palestine.