A Survey on Migration Process of Mobile Agent

Oyediran M. O, *Member, IAENG***,** Fagbola T. M, Olabiyisi S. O, *Member, IAENG*, Omidiora E. O, Fawole A. O *Member, IAENG*

Abstract- Mobile agent is a software object that migrates through many nodes of a heterogeneous network of computers under its own control in order to perform task using resources of these nodes, during the past several years, mobile agent has received significant attention. Not only in the wide range of commercial and law enforcement applications, but also the availability of feasible technologies, Although current migration process of mobile agent systems have reached a certain level of maturity, Autonomous migration of mobile agent is an area that has witnessed a lot of research activity and their development is still limited by the conditions brought about by many real applications. Several approaches have been proposed for the migration process of mobile agent. This paper provides an up-to-date survey of migration process of mobile agent research. To present a comprehensive survey, we categorize existing mobile agent migration process approaches and present detailed descriptions of representative methods within each category with special focus on the soft computing approaches.

Index Term - Mobile agent, Migration, Itinerary

I INTRODUCTION

In the past few years, computer systems have evolved from monolithic computing device to much more complex client-server environment (Osunade, 2007). Now, new phase of evaluation allows complete mobility of application code among supporting platforms to form a loosely coupled distributed system. Mobile-agent paradigm is one such technology. Mobile agents paradigm has captured researchers' and industry's attention long time ago because of its innovative capabilities and attractive applications (Preeti, Sattyam and Pragyan, 2012). Mobile agent is a software entity that migrates from one host to another, performing data collection and software configuration tasks on behalf of a user, it can migrate among computers and agent runtime environments (ARE). (Mohamed, khaoula and noreddine, 2012).

Manuscript received June 7, 2016; revised June 9, 2016.

Oyediran M. O **(**Member, IAENG**),** is with Department of computer science and engineering LAUTECH, Ogbomoso, Nigeria. (+2348038637607, mayor_yoppy02@yahoo.com). Fagbola T. M, is with Department of Computer Science FUOYE, Oye – Ekiti, Nigeria. Olabiyisi S. O **(**Member, IAENG), is with Department of computer science and engineering LAUTECH, Ogbomoso, Nigeria. Omidiora E. O, is with Department of computer science and engineering LAUTECH, Ogbomoso, Nigeria, Fawole A. O **(**Member, IAENG**)** is with Department of Electrical Engineering The Polytechnic Ibadan, Nigeria.

The following properties distinguish mobile agents from other computing programs:

- **Mobility-** Mobile agent has the ability to move from one host to another, either by moving the agent's code or by serializing both code and state to allow the agent to continue the execution in a new context.
- **Communication -** Mobile agent must have the ability to communicate with others agents of the system in order to exchange information and benefit from the knowledge and expertise of other agents.
- **Adaptability -** Mobility of agent required to learn about user's behavior and adapt it to suit the user. Indeed, to evolve adequately the differences between heterogeneous systems, the agent must be able to adopt the changes during the execution.
- **Autonomy-** Mobile agent must be able to make his own decision to be performed to achieve the user's tasks, also he must be able to migrate from one machine to another in the network and execute the user's tasks.
- **Persistence -** A persistent agent it will be able to retain the knowledge and state over extended period of time to be accessed later on. Once the mobile agent is set up, it is not dependent on system that has been initiated and it is automatically recovered when the agent is terminated or when it is flushed from memory to the database.

Mobile agent migration can be use instead of communications between a server-side and a client-side program. This enables to develop distributed systems, such as a ubiquitous computing environment, without being aware of communications APIs (Application Programming Interface) and protocols (Higashino, Takahashi, Kawamura and Sugahara 2012). MA have several advantages in distributed system (a) By migrating to the other side of an unreliable network link, an agent can continue executing even if the network link goes down, even this mobility property makes mobile agents particularly attractive in mobile computing environments, (b) By migrating to an information resource, an agent can invoke resource operations locally, eliminating the network transfer of intermediate data (Soheil, Mohammad, Behzad and Ahmad). However, this paper, reviewed the current development of mobile agent migration process. It is organized as follows: Section 2 review related works on migration process. In section 3 materials and methods (comparative evaluation metrics) of mobile agent. Section 4 results, and a summary and discussion of open problems and unaddressed research issues are presented in Section 5.

ISBN: 978-988-14047-1-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

II LITERATURE REVIEW

One of the characteristics of mobile agents is the ability to transfer from host to hosts over a network, migration of mobile agent means that the movement of an agent to another location in the network (e.g. computer) and transparent continuation at the point before the migration occurred, several research have been done on mobile agent migration process, Garima and Prakash, 2012, presented mobility and migration pattern of mobile agent, in the paper, migration pattern and migration strategies were discussed, he identify that there are various migration strategies. One possibility is to send the complete *program* (whole code) over the network. The opposite is to transmit only certain required parts (units) of the code. Another choice is whether to push code over the network, i. e. code will be sent over the network in advance, or to pull (download) code from a reachable location, i. e. the mobile agent (the execution unit) loads code from some suitable source. If the push code variant is used, code could be sent to the next location only, or to all locations on the agent's itinerary. One of the pitfalls of the "push strategy" is that it drives classes that could not have been used in the next locations or could never have been used at all. On the other hand, a "pull strategy" requires a fast reliable retained connection or at least a fast way to reconnect to the agent source through the agent lifetime (Xu and Qi, 2008). All of these strategies can be classified as follows:

Push-all-to-next

The code and all referenced objects are totally transferred to the next location

Push-all-to-all

The complete code of the agent are transmitted to all destination platforms the agent intending to visit so it needs all itinerary to be known in advance.

Pull-all-units

The agent only transmits the data and after the destination receives it starts to download all class files immediately when the first class file must be downloaded.

Pull-per-unit

After the destination receives the data, it tries to download the needed class file only.

Framework from mobile agent migration were also discussed, the typical behavior of a mobile agent is to migrate from one site to another from time to time. During the process of migration, the current site, i.e. the one the agent currently resides on, is called the Home platform and the other site to which the agent wants to migrate to is called the Destination Platform. The sender and the receiver must communicate over the network and exchange data about the agent that wants to migrate. Thus, it was said that some kind of communication protocol is driven, and they call that the migration protocol (Peine, 2002). Some systems simplify this task to an asynchronous communication, comparable to sending an electronic mail, whereas other systems develop rather complicated network protocols on top of TCP/IP. The migration process contains six steps, which are executed in sequence. The first three steps are executed on the home platform:

1) Initialize the migration process and suspend thread: The process of migration typically starts with a special command, the migration command, by which the agent announces its intention to migrate to another site, whose name is given as parameter of the migration command. The first task for the site is now to suspend the execution thread of the agent and to guarantee that no other child thread is still alive. This requirement is important for the next step, where it is imperative that data and state are frozen and cannot be modified later on.

2) Capture agent's data and execution state: The current state of all variables (the data) of the agent is serialized, i.e. their current values are written to an external persistent representation, e.g. a memory block or a file. The agent's state is also stored there, so that the point of suspension is known. Result of the serialization process is the serialized agent which is a flat byte stream that consists of the agent's data and state information.

3) Transfer the agent: The serialized agent is transferred to the destination platform using a migration protocol. Whether any code is sent to the Destination platform depends on different parameters.

The last three steps (4-6) are executed on the Destination platform.

4) Receive the agent: The serialized agent is received using the migration protocol. The destination platform checks whether the agent can be accepted based on information about the agent's owner and the home platform. The destination platform may filter out agents that come from sites that are unknown or not trusted.

5) Desterilize the agent: The serialized agent is desterilized, i.e. the variables and the execution state are restored from the serialized agent. The result of this step should be an exact copy of the agent that existed on the home platform just before reaching the migration command.

6) Start agent execution in new thread: The destination platform resumes agent execution by starting a new thread of control. At least when resuming execution, the agent's code is needed. In this general framework we make no assumption about how the code is transferred to the destination platform. One possible technique is for example that the destination platform loads the code from the agent's home site or its code server.

Preeti, Sattyam and Pragyan, 2012 developed a secure migration process for mobile agents from one host to another host named Mobile-C The migration process of mobile agents and ACL messages in Mobile-C is inspired from the SSH protocol. The security protocol was based on SSH because SSH already contains the key features required for our security process. The SSH protocol provides CIA system for the transfer of data without utilizing a central server. In his work for authentication, integrity and confidentiality he use several Algorithms. They are as follow:

Authentication process

Authentication refers to a process in which an agency ensures that the other agency in a conversation is in fact who it is declared to be. Before the secure transfer of a mobile agent between two agencies, they must authenticate each other. Each agency in a network contains a list of known hosts provided by the administrator (Xu and Qi, 2008). This list provides RSA public keys of other trusted agencies in a network. Before agency *A* wants to transfer a mobile agent to agency *B*,

ISBN: 978-988-14047-1-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

agency *A* must verify that agency *B* contains a correct private key for the public key in agency *A*'s known-host list. In addition, agency *B* must verify that agency *A* contains a correct private key for the public key in agency *B*'s known-host list.

Confidentiality

ISO defines confidentiality as '*ensuring that information is accessible only to those authorized to have access*es'. This means that while a mobile agent is migrating from agency *A* to agency *B*, it would not be accessible in an understandable form by any adversary. Mobile-C uses an AES 256 bit key to encrypt the mobile agent at the sending agency and to decrypt it at receiver agency. The insurance of security of this process relies on a secure transfer of the AES 256 bit key. Public key en-/decryption is used to transfer the AES key securely. The AES key is exchanged between two agencies in the authentication process (Tentori, Rodrıguez and Vara, 2011). This eliminates the further exchange of messages between two agencies for the AES key transfer. After successful authentication, the sender agency encrypts the mobile agent with the AES key and the receiver can decrypt it. According to National Institute of Standards and Technology (NIST) AES with 256 bit key size is safe to use for data encryption until 2030. The same nonce (as used in the authentication process) is used as session identifier during the transfer of both the AES key and the mobile agent. This is to avoid the replay back attack on agencies.

Integrity

Integrity is a process to ensure that the contents of a mobile agent are the same as sent by the sender agency. In other words, a successful integrity check should ensure that the agent was not tampered with while in transit from the sending agency to the receiving agency. Mobile-
C uses a SHA2 hash code to check the integrity of mobile agents.

Random number generation

True random number generation is always an issue and an important concern for cryptographic algorithms. The programming language C's random function has vulnerabilities and is thus not recommended for use in cryptographic applications. For this reason, Mobile-C uses Hardware.

Volatile Entropy Gathering and Expansion (HAVEGE) for high random number generation.

It is a heuristic software approach to generate empirically strong random numbers (Peine, 2002). Mobile-C uses HAVEGE to generate the nonce, challenge text, and AES 256 bit key during the mobile agent migration process.

Mohamed, Khaoula and Noreddine, in the paper they present the design and the implementation of Mobile-C, an IEEE Foundation for Intelligent Physical Agents (FIPA) compliant agent platform for mobile C/C++ agents. Such compliance ensures the interoperability between a Mobile-C agent and other agents from heterogeneous FIPA compliant mobile agent platforms. Also, the Mobile-C library was designed to support synchronization in order to protect shared resources and provide a way of deterministically timing the execution of mobile agents and threads. The new contribution of this work is to combine the mechanisms of agent migration and their synchronization.

Higashino, Takahashi, Kawamura, and Sugahara, 2012 propose a cache mechanism for mobile agent migration. They focus on an agent runtime environment and try to reduce data traffic in mobile agent migrations. In the proposal, an agent runtime environment caches agent codes and agent status. Cached codes and status are reuse when a mobile agent comes back again. Thus, the method enables to reduce data traffics caused by mobile agent migration at the agent runtime environment level. The method also allows flexible implementations of mobile agents, since an agent runtime environment is independent from the mobile agent behaviors. The method was applied on a mobile agent framework, called Maglog, and conducted experimental results.

Rani and Batra, in 2014, perform a comparison of migration strategy of mobile agent system, according to the paper, there exist three migration strategies.

(A)Ideal Migration Strategy: it is also called static migration. In this approach, a explicit itinerary graph is created based on its initial itinerary, after creating itinerary graph load information of itinerary graph is fetched. The drawback of this approach is that the change of load will outdate the edge and it requires a centralize server.

(B)One step Migration: This approach finds an optimal solution step by step during migration. It considers only the load on adjacent vertex. The drawback of this approach is that it does not provide the global optimal solution.

(C)Learning Migration Strategy: This approach tries to find out a globally optimal solution, in this approach during each time, agents records the software and hardware load information of a network. Next time, agent can use those travel experiences accumulated during previous migration. Each time, the old experiences will be updated by new information collected by agent. Rani and Batra, in 2014 also discuss, classification of itinerary, types of itinerary and itinerary planning of mobile agent.

According to Rani and Batra, Itinerary is the set of sites that MA has to visit, it can be classified as:

1. Static
2. Dynamic

1) Static planning: When the itinerary is fixed, the agent migration path is known at the MA's creation time and it does not change during the agent's execution. For instance, fixed itinerary agents are suitable for visiting a predetermined list of devices to collect data, where the itinerary is typically supplied by the user.

2) Dynamic planning: In dynamic planning, the mobile agent autonomously determines the source nodes to be visited and the route of migration according to the current network status.

Types of Itinerary;

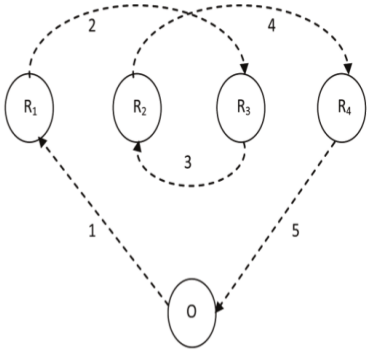
1. Static Itinerary Static Order (SISO)
2. Static Itinerary Dynamic Order (SIDO)
3. Dynamic Itinerary Dynamic Order (DIDO)

If we know the different server address then it is best to use the static itinerary agent. If we do not know the server address then it is better to use the dynamic itinerary agent.

1. **Static Itinerary Static Order (SISO)** In static itinerary the list of the remote host address is given by the owner at the time of dispatching the agent, the mobile agent should visit only the listed remote host and return to its

ISBN: 978-988-14047-1-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

home. In SISO, the agent should visit only the given remote host in the given order.



Static Itinerary	Static Order of Visit
R_1	1
R_2	3
R_3	2
R_4	4

Fig 2: Static Itinerary Static Order

2. **Static Itinerary Dynamic Order (SIDO)** In SIDO, the agent should visit only the given list of remote host in the dynamic order. The order of visiting the remote host is decided based on the current conditions (shortest path or network traffic based routing) of the host, where the agent is currently residing.

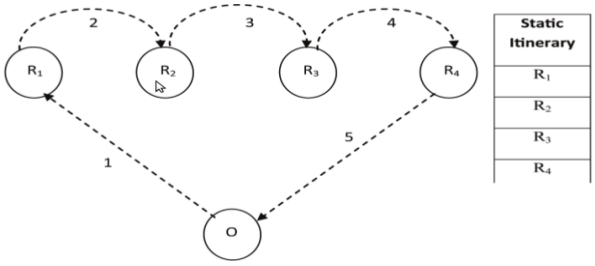


Fig. 3 Static Itinerary Dynamic Order

3. **Dynamic Itinerary dynamic Order (DIDO)** In DIDO mobile agent, the client only knows the very first remote server for sending the mobile agent whether the required content is there or not. The remaining remote server will be visited with the help of the server where the agent is currently residing that is based on the requirement of the agent. DIDO mobile agent will roam with two different things.

1. Query Based
2. Non Query Based

1. Query Based Approach In query based approach the client will send the mobile agent with the query for the particular content to the first remote server. The first remote server will be selected with the name. If the content is there than agent will collect the information otherwise it will not get any information and ask the help of the server to transfer it to the next server.

2. Non Query Based Approach In Non query based approach it will ask the server to forward it to the next server which is having the information relevant to the query (in case of query based) otherwise it will ask the server to forward to the nearby servers. After visiting the particular number of servers (decided by the client), agent will return back to the home (client) with the information.

III MATERIALS AND METHOD (Comparative evaluation metrics)

It was recommended to use a standard evaluation metric to benchmark any developed migration process in mobile agent, table 1, list some famous evaluation metrics.

IV RESULTS, DISCUSSION AND REMARKS

In this section, the results from the aforementioned evaluation metrics were discussed. Rani and Batra (2014) in his paper, presented his result as follows: When the size of MA code is varied from 7.5K to 160K, the freezing time of caching-added scheme is lower than the freezing time of without caching scheme. As the size of MA code increases, the performance of the scheme is improved from 118% to 125%, which was calculated by adding the freezing time of all the nodes or sites visited by the MA, Zhong Zhighui (2011) developed an ant colony algorithm based on path planning for mobile agent migration the, improved ant colony algorithm converges much faster than the basic ant algorithm and ant colony algorithm, the convergence of the cycles were increased by 28% and 20%. Convergence speed can not only provide better quality solutions, but also reduce the time to complete the task.

Higashino, Takahashi, Kawamura and Sugahara (2012), developed mobile agent migration based on code caching, and uses migration time as evaluation metric, the migration time of the model *with cache* is about half of *without cache*. To be precise, it is 52% *without cache*, Osunade and Atanda in 2008, in his research work titled analysis of two mobile agent migration patterns, uses transmission time as evaluation metric The transmission time for the mobile agent to migrate from the home host to the first host on the itinerary is dependent on the execution time used to load all needed code units on that host; the agent data and state; and the transmission delay between the two hosts. The equation is as follows:

$$ETT = ETX * \frac{S}{B}$$

Where,
ETT - Expected Transmission Time
ETX - Expected Transmission Count
S – Average size of packets
B – Current link Bandwidth

The result shows that the model performs better than the existing model in terms of transmission time.

V CONCLUSION

In this paper, we presented some major issues on migration process, the likes of: migration pattern, migration strategies, security aspect of mobile agent, classification and types of itinerary of mobile agent, evaluation metrics of mobile agent and results from different evaluation metric, were all discussed and reviewed. In summary, we present a comprehensive survey on migration process of mobile agent. We have tried our best to provide researchers in the field with the up-to-date information of research on migration process of mobile agent.

ISBN: 978-988-14047-1-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table 1: Evaluation metrics of mobile agent

AUTHOR/ YEAR	TITLE	EVALUATION METRIC
Rani and Batra (2014)	A Comparison of Migration Strategy of Mobile Agent System	Freezing time of mobile agent
Zhong Zhighui (2011)	Ant Colony Algorithm Based on Path Planning for Mobile Agent Migration	Convergence speed
Higashino, Takahashi, Kawamura and Sugahara (2012)	Mobile Agent Migration Based on Code Caching	Migration time of mobile agent
Osunade and Atanda (2008)	Analysis of two mobile agent migration patterns	Transmission time of mobile agent
Osunade (2007)	Data migration patterns for java-based mobile agent systems	Transmission time and network load of mobile agent
Oyediran *et al.,* (2016)	Development of an optimized mobile agent migration pattern for pull-all data strategy	Transmission time and network load of mobile agent

REFERENCES

[1] Mohamed B, Khaoula A. and Noreddine G: "Communication and migration of an embeddable mobile agent platform supporting runtime code mobility" International Journal of Advanced Computer Science and Applications, 3(1), 50-56. 2012.

[2] Osunade: "Data migration patterns for java based mobile agent system" {unpublished Ph.D Thesis} in the department of computer science, university of Ibadan, 1-80. 2007.

[3] Osunade S. and Atanda F.A.: "Analysis of two mobile agent migration patterns", journal of mobile communication, 2(2), 64-72. 2008.

[4] Rani S., Batra E. S: "A comparison of migration strategy of mobile agent system", International Journal of Advanced Research in Computer Science and Software Engineering. 4(9), 117-122. 2014.

[5] Zhong Zhishui: "Ant Colony Algorithm Based on Path Planning for Mobile Agent Migration", Elsevier Science Direct. 23, 1-8. 2011.

[6] Higashino M., Takahashi K., Kawamura T., and Sugahara K: "Mobile agent migration based on code caching", 26th International Conference on Advanced Information Networking and Applications Workshops, 651 – 656. 2012.

[7] Preeti S., Sattyam K. M., and Pragyan V: "A secure migration process for mobile agents form one host to another host", Journal of computer Applications, 5(4), 117 – 123. 2012.

[8] Garima verma, Atma Prakash Singh: "Mobility and migration pattern of mobile agent" Azad Institute of engineering & Technology, Lucknow, India, 1 – 5. 2012.

[9] Soheil J., Mohammad H., Behzad M and Ahmad K: "Clone-based mobile agent itinerary planning using separate trees for data fusion in WSNS", International Journal of Wireless & Mobile Networks, 4(4), 228 – 244. 2012.

[10] Bellifemine F., Caire G., Poggi A., Rimassa G: "A software framework for developing Multi - agent applications", Lessons learned. Information and Software Technology; 50(1–2): 10–21. 2008.

[11] Gray R., Cybenko G., Kotz D., Peterson R., Rus D: "Applications and performance of a mobile-agent system" Software—Practice and Experience; 32(6):543 573. 2002.

[12] Peine H: "Application and programming experience with the Ara mobile agent system". Software—Practice and Experience 2002; 32(6):515-541. 2002.

[13] Kawamura T., Hamada Y., Sugahara K., Kagemoto K., and Motomura S: "Multi-agent-based approach for meeting scheduling system," Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi - Agent Systems, pp. 79–84. 2007.

[14] Jurasovic K., Jezic G., and Kusek M: "A performance analysis of multi-agent systems,"International Transactions on Systems Science and Applications, vol. 1, no. 4, pp. 335–342. 2006.

[15] Xu Y. and Qi H: "Mobile agent migration modeling and design for target tracking in Wireless sensor networks," Ad Hoc Networks, vol. 6, no. 1, pp. 1–16. 2008.

[16] Wu Q., Rao N.S.V., Barhen J: "On computing mobile agent routes for data fusion in distributed sensor networks," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 6, pp. 740–753. 2004.

[17] Zhirou Z. and Wengang C: "Model and optimization of mobile agent's migration in grid," Proceedings of the 2nd International Conference on Bio-Inspired Computing: Theories and Applications (BICTA '07), pp. 127–130. 2007.

[18] Tentori M., Rodr´ıguez M., and Favela Vara J: "An agent-based middleware for the design of activity-aware applications," IEEE Intelligent Systems, vol. 26, no. 3, pp. 15–23. 2011.

[19] Al Shrouf F, Abusaimeh H, Al Shqeerat K, Al Omari M: "Evaluating time performance optimization analysis for mobile agent message communication using assignment computing agent". Journal of Applied Sciences; 12(12):1290-1296. 2012.