Journal of Mobile Communication 2 (2): 64-72, 2008
ISSN: 1990-794X

Analysis of Two Mobile Agent Data Migration Patterns

S. Osunade and F.A. Atanda
Department of Computer Science, University of Ibadan, Ibadan, Nigeria

Abstract: Mobile agents has found increasing used in the development of applications for distributed systems such as the Internet based on the premise that it often reduces network load and improves performance as compared to other design paradigms as client-server. However, studies have shown that mobile agent often performs poorly as compared to client-server. The poor performance of mobile agent is attributable to its too simple existing migration pattern. In this study, the migration characteristics of a single agent system are analyzed when the migration pattern for the data collected on each host is different. Mathematical models were developed for each stage of the migration process and programmed in Java, new migration pattern is developed and compared with the existing pattern for situations when data is migrated or not migrated. The results show that the transmission time is better for the new data migration pattern when an agent migrates with data.

Key words: Mobile agent, data, migration, transmission time

INTRODUCTION

A mobile agent is a program that can migrate from a starting host to many other hosts in a network of heterogeneous computer systems and fulfill a task specified by its owner. It works autonomously and communicates with other agents and host systems. During the self-initiated migration, the agent carries all its code and data and the complete execution state with it.

Migration, which is the transfer of a mobile agent from one host to another host, is a typical behaviour in a mobile agent system. Braun *et al.* (2001, 2005) described the migration process as follows:

> "When a mobile agent decides to migrate to another host the underlying agency is responsible to stop the agent execution, serialize the agent and to transfer the agent's state to the destination agency via network connection. The time it takes for the migration process, called transmission time, is influenced by network bandwidth and network latency. The destination agency receives and deserializes the agent's state and then re-starts the agent by creating a new thread and invoking a specific method."

An important technical argument in favor of mobile agents is the network load argument (sometimes also called performance argument). According to this argument, mobile agents are able to save network load and therefore, decrease execution time to some extent by shipping code close to the data instead of shipping data to the code as it is done in the client-server paradigm.

In executing its migration process, the data collected from a host is accumulated by the mobile agent in Java-based mobile agent systems i.e. the mobile agent adds the data collected to its existing data and carries it along until it returns to the originating host. This poses a number of problems such as loss of data, termination of the mobile agent before it reaches the originating host, long return time for the mobile agent if its data size is large and the delay in waiting for the mobile agent to visit all the hosts on its itinerary.

The growth and advantages offered by Internet services requires applications that reduce network traffic, traffic congestion, transmission time and bandwidth utilization. Existing mobile agent systems offer excellent operational performance only under certain conditions (Braun, 2003), which has impeded the adoption of mobile agents in most distributed applications.

Braun *et al.* (2001) identified several points during the migration process where performance can be improved and divided them into two classes: transmission aspects and runtime aspects. All techniques that influence network load and transmission time during agent migration are grouped under transmission aspects. Runtime aspects involve the techniques by which an agent's execution time can be improved.

This study focuses on the transmission aspects of Java-based mobile agent systems especially on the performance effects that result when the data collected by the mobile agent on each host on its itinerary is sent back to its home agency before migrating to the next host on its itinerary using a different pattern.

Corresponding Author: S. Osunade, Department of Computer Science, University of Ibadan, Ibadan, Nigeria

Literature review: The benefit of mobile agents in the development of distributed systems is the reduction of network load when compared to the other design paradigms. This is done by moving the mobile agent code close to the data instead of moving the data to the code as it is done in the client-server paradigm. This benefit has been validated by experiment. However, there are cases where mobile agents perform less than client-server techniques such as in information retrieval tasks where the bandwidth is high. In order to correct this anomaly two approaches have been proposed from literature.

The first approach requires that a static decision be taken on which paradigm: client-server, remote evaluation, code on demand and mobile agents, to use in the development of an application based on mathematical analysis. Carzaniga *et al.* (1997) and Vigna (1998) developed a simple performance model for the information retrieval domain, while Baldi *et al.* (1997), Picco (1998) and Baldi and Picco (1998) used this approach for an application from the network management domain. The authors using this approach concluded that no paradigm is better than the others for every application scenario but the choice of the paradigm to exploit must be performed on a case-by-case basis, according to the specific type of application and to the particular functionality being designed within the application (Vigna, 1998). They found that mobile agents produce the highest network traffic compared to all other design paradigms. This is because an agent carries all documents already found, whereas on all the other paradigms documents are sent back to the client immediately. Thus, a mobile agent's data grows continuously with each host visited, so that in sum it grows quadratically as the number of servers.

The second approach proposed by Straâer and Schwehn (1997) was predicated on the notion that it is only the mixture of agent migrations and remote procedure calls leads to minimal network traffic. They developed a mathematical model for measuring the performance based on network load and execution time of the traditional client-server remote procedure call (RPC) method and mobile agents for single and multi interactions for a given application scenario. The model required several parameters such as the amount of communication necessary between client and server(s) as well as bandwidth and latency for all network connections to be known in advance. The conclusions were that single agent migration had advantages compared to a single remote procedure call when the amount of data to be processed is large, compared to the size of the agent. However, for multiple interactions an alternating sequence of remote procedure calls and agent migrations performed better than a pure sequence of remote procedure calls or a sequence of agent migrations. The results from the model show that a mixture of techniques produces minimal network load. They validated their findings by experimentation using Mole-a mobile agent system. Chia and Kannapan (1996) also reached the same conclusion when 3 different mobile agents migration patterns were compared.

The second approach solved the problem of deciding which paradigm to use by combining agent migration with remote procedure calls to reduce network load. The mobile agent only migrates to some of the hosts on the network while the other hosts are accessed using remote procedure calls. The optimal sequence depends on the size of requests and results and on the network quality between each pair of hosts (Braun, 2003). Iqbal *et al.* (1998) presented several algorithms to compute the optimal migration sequence of a single agent. Their approach is based on an algorithm to determine the shortest path in a directed and weighed graph.

Due to the fact that static approaches are not sufficient for all kinds of application and the drawbacks of the mixture approach include the assumption of the values of several parameters such as network bandwidth, latency, request and result size, makes this approaches undependable. Hence, this study focuses on the first approach of mobile agent migration with the aim developing a new migration pattern so as to reduce network load using mathematical analysis.

Data migration patterns: The following represents the two migration patterns (existing and developed) (Fig. 1) for the data collected by a mobile agent:

Existing pattern: The mobile agent can migrate from host to host on its itinerary with the data generated on each host. For example, the mobile agent moves from the home host A without any data generated to host B. At host B, data is generated through the activities of the mobile agent. The mobile agent migrates to the next host C with the mobile agent and the data generated at host B. This process is repeated until it returns to the home host A. This is the default data migration pattern used by Java-based mobile agent systems.

Developed pattern: The mobile agent can migrate from host to host on its itinerary without the data generated on each host. For example, the mobile agent moves from the home host A without any data generated to host B. At host B, data is generated through the activities of the mobile agent. The mobile agent transmits the data

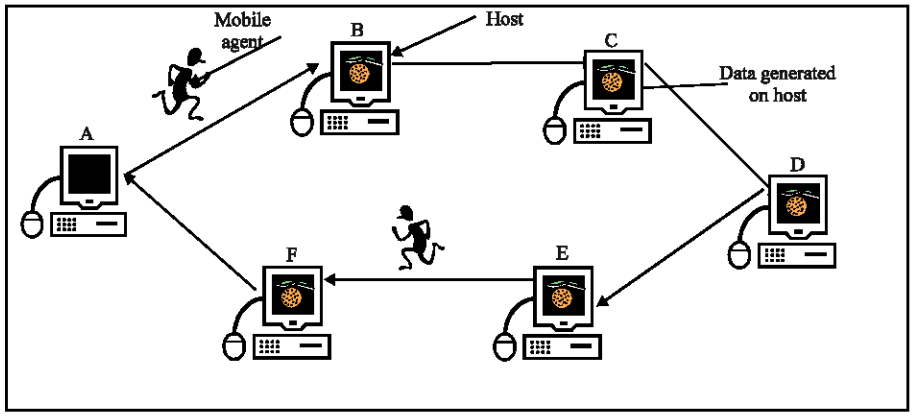

Fig. 1: Conceptual framework of the reasearch

generated to the home host A before migrating to the next host C. This process is repeated until the mobile agent returns to the home host A.

MATERIALS AND METHODS

This research used mathematical formalism to analyze the performance of four mobile agents' migration strategies when the data collected on each host is treated differently.

The mathematical model of the existing migration pattern as developed by Braun (2001) was modified to obtain an equivalent model for the developed migration pattern. Introduction of compression factor (σ) to both migration patterns is part of the modification. Each migration strategy had 3 equations for each stage of the migration process.

Java simulation program was developed and used in simulating the mathematical models based on data from table 1.0. Analysis and comparison of the migration patters was based on the output of the simulation program that was plotted on graph using Microsoft Excel.

Models development: This model used the network load (B) and transmission time (T) as the metrics for performance evaluation to an ordered set of hosts (servers), represented as $L = L_1,\ldots\ldots L_m$. The agent consists of some class files which can be dynamically loaded during execution from the agent's home host. The decision on which class files must be loaded is influenced by the communication of the agent to the local agent server.

Network load: In modelling, the network load (B), the following assumptions have been made:

- Agent has to visit m servers {L1, . . . ,Lm} to collect data from each server.
- An agent consists of n units of code or n class files.
- Each unit of code has a length or size B_c^k, k =1,…n.
- The sum of all code units is $B_c = \Sigma B_c^k$, k = 1,…n.
- An agent has a data of length or size B_d.
- An agent's state information is of length or size B_s.
- A request to load a specific code unit has a length or size B_r.
- The probability of dynamically loading code unit k on a host is P_{Li}^k.
- On a host the agent's data increases by d_L bytes.
- The migration is from L_a to L_{a+1} where a = 1,…m-1 is a host on the itinerary that is not the home host.

The migration process consists of marshalling data and state, transmitting data, state and code to the destination host and unmarshalling of data and state information.

Transmission time: To model the roundtrip time (T), the following simplifications are assumed:

- Marshalling and unmarshalling of data on each host is linear in time and modelled by μ: IN → IR.
- For each pair of platforms we know throughput τ: L × L → IR.
- For each pair of platforms we know delay δ: L × L → IR in advance.
- σ is the compression factor which ranges between 0-1.
- S represents various strategies i.e. push-to-next, push-to-all, pull-code-per-unit and pull-all-code.

The migration process is divided into three steps. First, the agent migrates from its home agency (T_{lve}) L_0 to the first server L_1 of the given itinerary. Second, the agent

migrates from server (T_{mig}) Li to server L_{i+1}, where i = 1. . . m-1. Last, the agent migrates (T_{home}) back to its home agency. With this information the equation for the network load and transmission time for the two migration patterns are presented.

Existing data migration pattern: The existing data migration pattern is the default data migration pattern implemented by most mobile agent systems. The mobile agent code and data collected are migrated using the same migration strategy to the next or all host (s) based on the agent's itinerary. Thus the mobile agent increases in size as it moves from host to host. The network load and transmission time for each migration strategy have been written as a set of 3 models. The transmission time is a combination of the marshalling factor, the delay in transmission from one host to the next and the rate at which the network load is migrated to the next host.

Push-to-next-host migration strategy: In this migration strategy, all implementation code and data is moved to the next host on the itinerary of the mobile agent.
The equations for network load are:

$$B_{lve}(L,S) = B_c + B_d + B_s \quad (1)$$

$$B_{mig}(L,a,S) = B_c + B_d + \sum_{i=1,..a} d_{Li}(1-\sigma) + B_s \quad (2)$$

$$B_{home}(L,S) = B_d + \sum_{i=1,..,m} d_{Li}(1-\sigma) + B_s \quad (3)$$

The equations for the transmission time are:

$$T_{lve}(L,S) = 2\mu(B_d + B_s) + \delta(L_h, L_1) + \frac{B_{lve}(L,S)}{\tau(L_h, L_1)} \quad (4)$$

$$T_{mig}(L,a,S) = 2\mu(B_d + B_s + \sum_{i=1,..,a} d_{Li}(1-\sigma)) + \delta(L_a, L_{a+1}) + \frac{B_{mig}(L,a,S)}{\tau(L_a, L_{a+1})} \quad (5)$$

$$T_{home}(L,S) = 2\mu(B_{home}(L,S) + \delta(L_m, L_h) + \frac{B_{home}(L,S)}{\tau(L_m, L_h)} \quad (6)$$

Push-to-all-hosts migration strategy: All implementation code to be migrated is sent to all the hosts on the itinerary before the mobile agent migrates. As it migrates from host to host it carries the data along with it but not the code.
The equations for network load are:

$$B_{lve}(L,S) = |L| B_c + B_d + B_s \quad (7)$$

$$B_{mig}(L,S) = B_d + \sum_{i=1,..,a} d_{Li}(1-\sigma) + B_s \quad (8)$$

$$B_{home}(L,S) = B_d + \sum_{i=1,..,m} d_{Li}(1-\sigma) + B_s \quad (9)$$

The equations for the transmission time are:

$$T_{lve}(L,S) = 2\mu(B_d + B_s) + \sum_{l=1,..,m} (\delta(L_h, L_l) + \frac{Bc}{\tau(L_h, L_l)}) + \frac{B_d + B_s}{\tau(L_h, L_1)} \quad (10)$$

$$T_{mig}(L,a,S) = 2\mu(B_{mig}(L,a,S) + \delta(L_a, L_{a+1}) + \frac{B_{mig}(L,a,S)}{\tau(L_a, L_{a+1})} \quad (11)$$

Same as Eq. 6.

Pull-all-code-units migration strategy: For this strategy, the mobile agent migrates to a host then requests for all implementation code from a code server or agent server before execution. The code is usually a jar file for Java-based mobile agent systems. On migration to the next host with data, it sends the same request for code again. The equations for network load are:

$$B_{lve}(L,S) = B_c + B_r + B_d + B_s \quad (12)$$

$$B_{mig}(L,a,S) = B_d + \sum_{i=1,..,a} d_{Li}(1-\sigma) + B_s + B_r + B_c \quad (13)$$

$$B_{home}(L,S) = B_d + \sum_{i=1,..,m} d_{Li}(1-\sigma) + B_s \quad (14)$$

The equations for the transmission time are:

$$T_{lve} = 2\mu(B_d + B_s) + \delta(L_h, L_1) + \frac{B_d + B_s}{\tau(L_h, L_1)} + \delta(L_1, L_h) + \frac{B_r + B_c}{\tau(L_1, L_h)} \quad (15)$$

$$T_{mig}(L,a,S) = 2\mu(B_d + \sum_{i=1,..a} d_{Li}(1-\sigma) + B_s) + \delta(L_a, L_{a+1}) + \frac{B_d + \sum_{i=1,..a} d_{Li}(1-\sigma) + B_s}{\tau(L_a, L_{a+1})} + \delta(L_{a+1}, L_h) + \frac{B_r + B_c}{\delta(L_{a+1}, L_h)} \quad (16)$$

Same as Eq. 6.

Pull-code-per-unit migration strategy: This migration strategy requires the code to be in units or classes which are requested from a code server or agent server on a unit by unit basis by the mobile agent as required when it executes on a host.

The equations for network load are:

$$B_{lve}(L,S) = B_d + B_s + \sum_{k=1,..n} P_{Li}{}^{k}\left(B_r + B_c{}^{k}\right) \quad (17)$$

$$B_{mig}(L,S) = B_d + \sum_{i=1,..,a} d_{Li} + B_s + \sum P_{Li}\left(B_r + B_c{}^{k}\right) \quad (18)$$

$$B_{home}(L,S) = B_d + \sum_{i=1,..,m} d_{Li}(1-\sigma) + B_s \quad (19)$$

The equations for the transmission time are:

$$T_{lve}(L,S) = 2\mu(B_d + B_s) + \delta(L_h, L_1) + \frac{B_d + B_s}{\tau(L_h, L_1)} + \phi_1 \quad (20)$$

Where:

$$\varnothing = \sum_{k=1,..n} P_{Li}{}^{kk}\left(\delta\left(L_s, L_h\right)\right) + \frac{B_r + B_c{}^{k}}{\tau\left(L_s, L_h\right)}$$

$$T_{mig}(L,a,S) = 2\mu(B_d + B_s + \sum_{l=1,..,a} d_{Li}(1-\sigma)) + \delta(L_a, L_{a+1}) + \frac{B_d + B_s + \sum_{i=1,..,a} d_{Li}(1-\sigma)}{\tau(L_a, L_{a+1})} + \phi_{a+1} \quad (21)$$

Same as Eq. 6.

New data migration pattern: The development of the new data migration pattern is based on the need for faster access by applications and users to the data being collected by the mobile agent and the need to improve the transmission time of the mobile agents during migration. Having observed the major problem of mobile agents is that it carries along with it data collected on host on its itinerary, this models concentrates on improving the transmission time of the migration stage i.e., T_{mig}. The equation for network load and transmission time of this model is equivalent to the 2 stages ($_{lve}$ and $_{home}$) of the existing pattern. Hence the modification is majorly on second stage of the migration process (i.e., B_{mig} and T_{mig}).

The new data migration pattern represents the scenario when the mobile agent can migrate from host to host on its itinerary without the data collected on each host. For example, the mobile agent moves from the home host A without any data collected to host B. At host B, data is collected through the activities of the mobile agent. The mobile agent transmits the data generated to the home host A before migrating to the next host C. This process is repeated until the mobile agent returns to the home host A.

Push-to-next-host migration strategy: The mobile agent moves all the implementation code to the next host but sends the data collected on each host to the home host. The equations for network load and transmission time are given.

B_{lve} same as Eq. 1

$$B_{mig}(L,S) = B_c + B_d + B_s + d_L\left(1-\sigma\right) \quad (22)$$

$$B_{home}\left(L,S\right) = B_d + d_L\left(1-\sigma\right) + B_s \quad (23)$$

T_{lve} same as Eq. 4

$$T_{mig}(L,a,S) = 2\mu(B_d + B_s) + \delta(L_a, L_{a+1}) + \frac{B_{mig}(L,a,S)}{\tau(L_a, L_{a+1})} + (\frac{1}{\tau(L_a, L_{a+1})} + 2\mu)d_L\left(1-\sigma\right) \quad (24)$$

T_{home} same as Eq. 6.

Push-to-all-hosts migration strategy: All implementation code to be migrated is sent to all the hosts on the itinerary before the mobile agent migrates. However, as the mobile agent migrates from host to host it sends the data collected on each host to the home host. The equations for network load and transmission time are given.

B_{lve} same as Eq. 7

$$B_{mig}\left(L,a,S\right) = B_d + d_L\left(1-\sigma\right) + B_s \quad (25)$$

B_{home} same as Eq. 23

T_{lve} same as Eq. 10

$$T_{mig}(L,a,S) = 2\mu(B_{mig}(L,a,S) + \delta(L_a, L_{a+1}) + \frac{B_{mig}(L,a,S)}{\tau(L_a, L_{a+1})} + (\frac{1}{\tau(La, La+1)} + 2\mu)d_L(1-\sigma) \quad (26)$$

T_{home} same as Eq. 6

Pull-all-code-units migration strategy: For this migration strategy, the mobile agent migrates to a host then requests for all implementation code from a code server or agent server before execution. The data collected on each host is sent to the home host while the necessary codes are requested when the mobile agent migrates to another host. The equations for network load and transmission time.

B_{lve} same as Eq. 13

$$B_{mig}(L,a,S) = B_d + B_s + B_r + B_c + d_L(1-\sigma) \quad (27)$$

B_{home} same as Eq. 23
T_{lve} same as Eq. 16

$$T_{mig}(L,a,S) = 2\mu(B_d + B_s) + \delta(L_a, L_{a+1}) + \frac{B_d + B_s}{\tau(L_a, L_{a+1})} + \delta(L_{a+1}, L_h) + \frac{B_r + B_c}{\tau(L_{a+1}, L_h)} + \left(\frac{1}{\tau(La, La+1)} + 2\mu\right)d_L(1-\sigma) \quad (28)$$

T_{home} same as Eq. 6.

Pull-per-code-unit migration strategy: This migration strategy requires the code to be in units or classes which are requested from a code server or agent server on a unit by unit basis by the mobile agent as required when it executes on a host. The data collected on each host is sent to the home host. The equations for network load and transmission time.

B_{lve} same as Eq. 17

$$B_{mig}(L,a,S) = B_d + B_s + \sum_{k=1,..n} P_{Li}^{k}\left(B_r + B_c^{k}\right) + d_L(1-\sigma) \quad (29)$$

B_{home} same as Eq. 23
T_{lve} same as Eq. 20

$$T_{mig}(L,a,S) = 2\mu(B_d + B_s) + \delta(L_a, L_{a+1}) + \frac{B_d + B_s}{\tau(L_a, L_{a+1})} + \phi_{a+1} + \left(\frac{1}{\tau(La, La+1)} + 2\mu\right)d_L(1-\sigma) \quad (30)$$

T_{home} same as Eq. 6

EVALUATION

In evaluating the agent's performance based on the two-migration pattern using the java simulation program, following consideration were made. The agent contains seven classes, with difference code size and probabilities as shown in Table 1. The agent's initial code size Bc is 5 KB, initial state size Bs of 3 KB and increases its data d_L by 5 KB on each host and request for code Br = 5 kbyte. The agent has to visits ten hosts each having throughput τ = 100 byte/sec and delay δ = 5 ms with marshalling factor μ = 5 ms. The two models were simulated using two different values (i.e., 0.5 and 1) for the selectivity constant (compression factor σ). The output of the simulation program was plotted on graphs as shown on the Fig. 2-5.

Table 1: Code size and download probabilities

Code class	Probability	Codesize (KB)
1	0.0, 0.5, 1.0	10
2	0.0, 0.5, 1.0	15
3	0.0, 0.5, 1.0	15
4	0.0, 0.5, 1.0	15
5	0.0, 0.5, 1.0	15
6	0.0, 0.5, 1.0	15
7	0.0, 0.5, 1.0	15

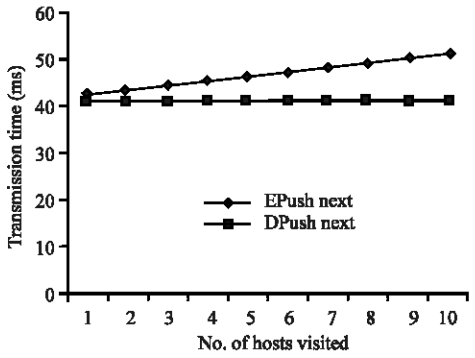

Fig. 2a: Push-to-next-host migration strategy

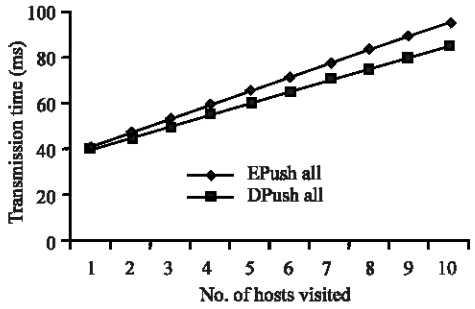

Fig. 2b: Push-to-all-host migration strategy

Transmission time when no data is migrated: This scenario depicts a situation where no data is collected on each host as the mobile agent migrates across the network. This represents the situation when the selectivity constant is 1 (σ = 1). This is presented graphically for each migration strategy below as the number of hosts visited increases.

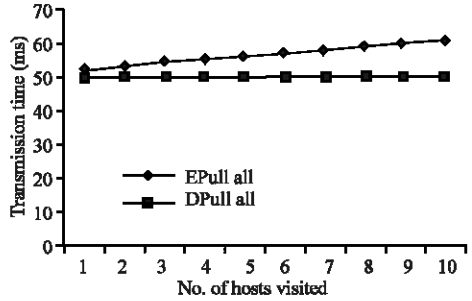

Fig. 3a: Pull-all-code-units migration strategy

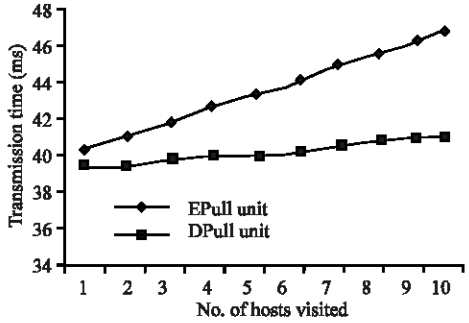

Fig. 3b: Pull-code-per-unit migration strategy

Figure 2a reveals that for push-to-next host migration strategy, the transmission time for existing pattern increases proportionally as the number of hosts visited increases. While the developed pattern has a constant transmission time irrespective of the number of host visited. It also reveals that the existing pattern has a higher transmission time than the developed pattern.

Figure 2b on the other hand, reveals that for push-to-all host migration strategy, the transmission time for both patterns increases proportionally as the number of hosts visited increases. It also, reveals that the existing pattern has a slightly higher transmission time than the developed pattern.

Figure 3a reveals that for pull-all-code-unit migration strategy, the transmission time for existing pattern increases proportionally as the number of hosts visited increases. While, the developed pattern has a constant transmission time irrespective of the number of host visited. It also, reveals that the existing pattern has a higher transmission time than the developed pattern.

Figure 3b on the other hand, reveals that for pull-code-per-unit migration strategy, the transmission time for both patterns increases proportionally as the number of hosts visited increases. It also, reveals that the existing pattern has a significant higher transmission time than the developed pattern.

Transmission time when full data is migrated: This scenario depicts a situation where the total data collected

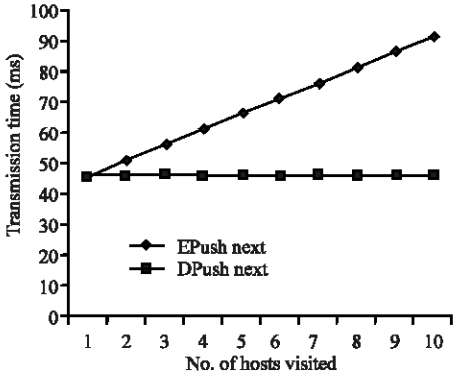

Fig. 4a: Push-to-next-host migration strategy

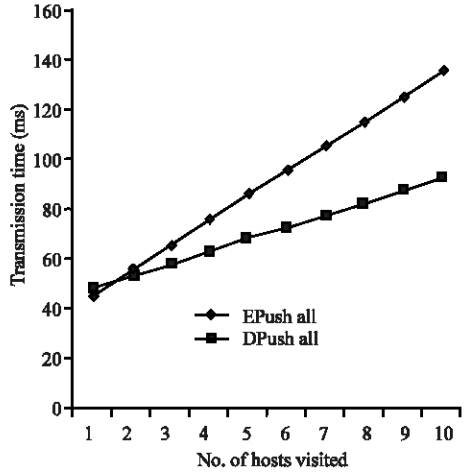

Fig. 4b: Push-to-all-host migration strategy

on each host was compressed to half its initial size before migrating with the mobile agent as it migrates across the network. This would represent the situation when the selectivity constant is 0.5 (i.e. $\sigma = 0.5$). This is presented graphically for each migration strategy below as the number of hosts visited increases.

In Fig. 4a, push-to-next-host strategy, there is a significant difference in transmission time for the existing and new data migration pattern as the number of hosts visited increases. While the developed data migration pattern has a lower and constant transmission for all hosts visited, the existing data migration pattern has an increasing transmission time.

In Fig. 4b, push-to-host-host strategy, there is a significant difference in transmission time for the existing and new data migration pattern as the number of hosts visited increases. Both the developed and existing data migration patterns have an increasing transmission time as the number of hosts visited increases. However, the developed migration pattern has a lower transmission time than the existing pattern.

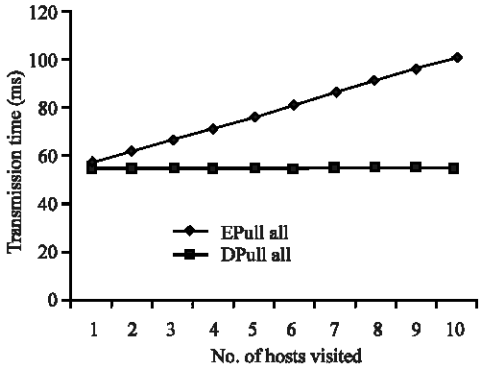

Fig. 5a: Pull-all-code-units migration strategy

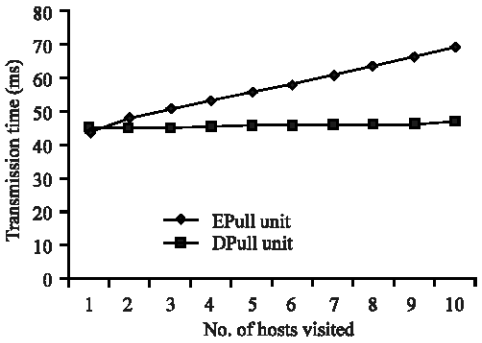

Fig. 5b: Pull-code-per-unit migration strategy

In Fig. 5a, pull-all-code-units migration strategy, there is a significant difference in transmission time for the existing and new data migration pattern as the number of hosts visited increases. While the developed data migration pattern has a lower and constant transmission for all hosts visited, the existing data migration pattern has an increasing transmission time.

In Fig. 5b, pull-code-per-unit migration strategy, there is a significant difference in transmission time for the existing and new data migration pattern as the number of hosts visited increases. While, the developed data migration pattern has a lower and constant transmission for all hosts visited, the existing data migration pattern has an increasing transmission time.

In both cases (i.e. σ =1 or = 5), Push-to-all-host migration strategy has the highest transmission time, follow by Pull-all-code-units and Push-to-next host migration strategies. Pull-code-per-unit migration strategy performs better than all other migration strategies in all situations by having the fastest transmission time.

CONCLUSION

This research concentrates mainly on the second stage of the migration process with the intent of reducing the data being carried by mobile agent along its itinerary as a way of improving the migration pattern which resulted in the development of new migration pattern. Results from the simulation of both models reveal that the developed model performs better than the existing model in all of the migration strategies. It is obvious from the result can be incorporated to any of the existing migration pattern of mobile agents systems so as to improve its performance and thereby returning mobile agent to its rightful place. It is hope that the developed models can still be improved upon further as a means of enhancing the performance of mobile agent.

REFERENCES

Baldi, M., S. Gai and G.P. Picco, 1997. Exploiting Code Mobility in Decentralized and Flexible Network Management. Proceedings of the First Workshop on Mobile Agents (MA), Germany K. Rothermel and R. Popescu-Zeletin, (Eds.). Lecture Notes in Computer Science, Pub. Springer-Verlag, 1219: 27-34.

Baldi, M. and G.P. Picco, 1998. Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications. Proceedings of the 20th International Conference on Software Engineering (ICSE), Japan R. Kemmerer and K. Futatsugi, (Eds.). IEEE Computer Society Press, pp: 146-155.

Braun, P., C. Erfurth and W. Rossak, 2000. An introduction to the Tracy mobile agent system. Technical Report Math/Inf/00/24. Institut fur Informatik, Friedrich-Schiller-Universitat. Retrieved ftp://ftp.minet.uni-jena.de/pub/ips/braun/bericht-00-24.pdf.

Braun, P., C. Erfurth and W. Rossak, 2001. Performance evaluation of various migration strategies for mobile agents.

Braun, P., 2003. The migration process of mobile agents: Implementation, classification and optimization. PhD Thesis. Friedrich-Schiller-Universitat Jena. xvii+293.

Braun, P., I. Mueller, R. Kowalczyk and S. Kern, 2005. Attacking the migration bottleneck of mobile agents, Technical Report: SUTICT-TR2005.01. http://www.it.swin.edu.au/centres/TechnicalReports/2005/SUTICT-TR2005.01.pdf.

Carzaniga, A., G.P. Picco and G. Vigna, 1997. Designing Distributed Applications with Mobile Code Paradigms. Taylor, R. (Ed.). Proc. 19th Int. Conf. Software Eng., pp: 22-32.

Chia, T.H. and S. Kannapan, 1996. Strategically Mobile Agents. In Proceedings of the First International Workshop on Mobile Agents (MA), Rothermel, K. and R. Popescu-Zeletin (Eds.). Pub. Springer-Verlag, Germany, 1219: 149-161. Citeseer.ist.psu.edu/chia96 strategically.html.

Iqbal, M.A., J. Baumann and M. Straßer, 1998. Efficient algorithms to find optimal agent migration strategies. http://elib.uni-stuttgart.de/opus/volltexte/1999/323/pdf/323.pdf.

Picco, G.P., 1998. Understanding, evaluating, formalizing and exploiting code mobility. PhD. Thesis. Politecnico Di Torino, Italy, xi: 206.

Straâer, M. and M. Schwehm, 1997. A performance model for mobile agent systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), USA. H.R. Arabnia, CSREA Press, 2: 1132-1140.

Vigna, G., 1998. Mobile code technologies, paradigms and applications. PhD. Thesis. Politecnico Di Milano, Italy, iii: 84.