Dispatching Mobile Agents with Secure Routes in Parallel

Yan Wang and Kian-Lee Tan

Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{ywang, tankl}@comp.nus.edu.sg

Abstract. In a distributed environment like the Internet, mobile agents can be employed to perform autonomous tasks such as searching and negotiating. However, for mobile agents to be widely accepted, performance and security issues on their use have to be addressed. In this paper, we propose a parallel dispatch model with secure route structures for protecting the dispatch routes of agents. This model facilitates efficient dispatching of agents in a hierarchical manner, and ensures route security by exposing minimal route information to hosts. To further enhance route robustness, we also propose a mechanism with substitute routes that can bypass temporarily unreachable hosts, using substitute hosts for deploying right dispatch branches and make later attempts to these failed hosts.

1 Introduction

In recent years, there have been increasing interests in deploying mobile agents carrying both code and data for distributed processing in an environment such as the Internet. For example, in electronic commerce (EC), a pool of mobile agents can be dispatched from a host to related e-shops to gather information, such as price, stock status, warranty and delivery service etc., for goods specified by a customer [1], [2], [3], [4]. Clearly, an efficient strategy is to dispatch a large number of agents to work in parallel [5], [6]. This will also provide customers with the possibility to find the "best" e-shop to make his/her purchases.

However, for mobile agent technologies to be accepted, performance and security issues on their use have to be addressed. First, to deploy a large number of agents require significant overhead to dispatch the agents. Novel methods for dispatching agents are desirable. Second, when a mobile agent arrives at a host for execution, the code and data will be exposed to the host and the resources at the host may also be exposed to the mobile agent. Thus, security mechanisms should be set up to protect mobile agents from malicious hosts as well as to protect hosts from malicious agents. Some works have been done to protect the hosts, e.g., the access privilege protocol [7], [8] and the role based mechanism [9] restrict an agent's access to resources of a host. Protecting the agent is also a difficult task. In particular, in EC environment, since e-shops are competitive, it is important to protect the routes of a mobile agent if it should visit a list of hosts (e-shops) or if it should dispatch other mobile agents to other hosts. If a malicious host knows the route information, it may tamper with it so

S. Qing, T. Okamoto, and J. Zhou (Eds.): ICICS 2001, LNCS 2229, pp. 386-397, 2001.

that its competitors that may offer better prices or services will not be visited. This calls for novel methods to be designed.

In this paper, we focus on the issues of efficiently dispatching mobile agents while protecting their routes. We first present a hierarchical dispatch model, which can efficiently dispatch a large number of mobile agents in parallel and is robust in the sense that an agent can be dispatched to any of the embedded hosts by delaying the trials to temporarily unreachable hosts. However, this comes at the cost of exposing all the addresses of descendent agents to hosts and hence it is not secure in the context of protecting mobile agents from malicious hosts. Based on this model, we present a security enhanced parallel dispatch model, which will not expose the information of all descendent agents except the children agents. Thus, we preserve the efficiency of the hierarchical model while ensuring routes security. In addition, we also give a solution to facilitate robustness without sacrificing on security and efficiency.

In this paper, we employ well-known cryptography technologies such as the asymmetric encryption algorithm, signature generating algorithm and X.509 authentication framework [10], [11]. In the following, we assume that there exists a secure environment including the generation, certification and distribution of public keys and each host can know the authentic public key of other hosts.

2 A Basic Security Enhanced Model for Parallel Dispatch

2.1 Binary Dispatch Model

In this paper, we assume an infrastructure where a set of marketplaces is connected to the Internet. Requests by users go through the agent A_{MSMA} running at the Master Server for Mobile Agents (MSMA), which is an execution environment for mobile agents. In MSMA, a customer agent can be created or dispatched. We call an agent a Worker Agent (WA) if its sole responsibility is to perform the tasks assigned to it, e.g., accessing data. If an agent also dispatches other agent besides performing the task of accessing data, it is called a Primary Worker Agent (PWA).

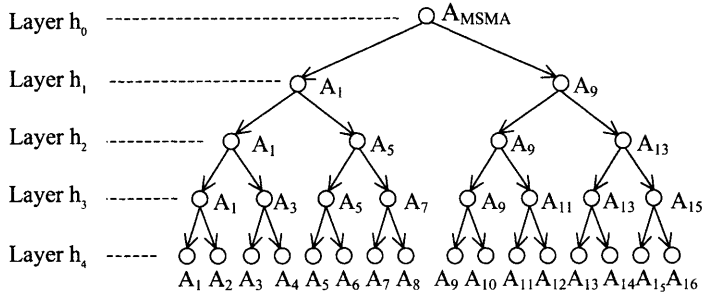

Fig. 1. Dispatch tree with 16 WAs

In this section, we introduce the proposed parallel dispatch model. For simplicity, we restrict our discussion to a *binary* dispatch model where an agent can dispatch two other agents resulting in a binary tree structure. Clearly, the model can be easily generalized to dispatch multiple (more than 2) agents. As shown in Figure 1, A_{MSMA} is responsible for dispatching PWAs and distributing tasks to them. Suppose A_{MSMA} has to dispatch 16 agents to different hosts. Now, they can be divided into 2 groups led by two PWAs, say A_1 and A_9. When agents A_1 and A_9 are dispatched, each of them has 8 members including itself. For A_1, it will dispatch A_5 and distribute 4 members to it. Then A_1 will transit to the same layer (i.e., h_2) as A_5, which is called a virtual dispatch. But now A_1 has 4 members only. Following the same process, A_1 will dispatch A_3 and A_2. At last, after all dispatch tasks have been completed, A_1 will become a WA and start its data-accessing task. In this model, in certain layer, a PWA can choose any of its members except itself to be the right child agent. In this way, any right branches can be surely deployed and any unreachable hosts can be bypassed to a later attempt. As a whole, since all PWAs are dispatched to different hosts, the dispatch process can be preformed in parallel. When there are $n=2^h$ mobile agents and Δt is the average time for dispatching a mobile agent, $(h+1)\Delta t$ will be the time for dispatching n mobile agents in the binary way. So, the dispatch complexity will be $O(\log n)$. Thus, the proposed model is both robust and efficient.

There are three alternative implementations for a PWA to create and dispatch a child agent in the IBM Aglet system [12]. The first approach is that the MSMA passes the child agent to the PWA who creates the child agent and encapsulates arguments such as the route and tasks and then dispatches it. This method is expected to be inefficient in a WAN environment. The second is to compress the framework of child agents to a .jar file and attach it to the PWA when it is dispatched. The child agent is created from the compressed file for being dispatched. The third one is to adopt the clone-like strategy. If some mobile agents have the same type of tasks, they can be put to the same group where a PWA can easily create a child agent by locally making a copy and modifying the static data. After encapsulating the route to the copy, the PWA can dispatch it to a remote host. A secure clone environment that provides security mechanisms to detect illegally forged agents is also an important issue that is out of the scope of this paper. The common feature for three alternatives is that arguments can be encapsulated to an agent when it is created. Here we address the secure dispatch route issue only with general-purpose models that can detect illegally forged agents, and do not restrict it to any implementation system.

2.2 Securing the Route Structure

In the basic binary dispatch model, to be robust, PWAs must expose all route information to the hosts. To ensure route security, we applied cryptographic technique to the model. To protect the routes, we should expose the addresses to a host only when necessary. For example, if an agent is at host A, and it has to dispatch an agent to host B, then the address of B must (obviously) be exposed to the host A; however, no other addresses should be exposed.

For the binary dispatch model, it is more complicated than traditional serial migration model since a PWA has different dispatch tasks in different layers. Only the

operations for a WA are simple. For the binary dispatch model, a basic definition of route structure, is as follows:

$$\begin{aligned} &(1)\quad \text{For a PWA at CH, } r(CH)=P_{CH}[PWA, ip(RH), r_L, r_R, \\ &\qquad S_{MSMA}(PWA, ip(PH), ip(CH), ip(RH), r_L, r_R, t)] \\ &(2)\quad \text{For a WA at CH, } r(CH)=P_{CH}[WA, ip(MSMA), \\ &\qquad S_{MSMA}(WA, ip(PH), ip(CH), ip(MSMA), t)] \end{aligned} \quad \textbf{(1)}$$

Where r(CH) denotes the route structure at the current host, CH, where the agent should go; ip(H) denotes the IP address of host H; RH and PH denote the right child's host and the parent host respectively; r_L and r_R denote the encrypted route for the left and right children respectively; $P_{CH}[M]$ denotes the message M is encrypted by the public key of the current host CH; and $S_{MSMA}(D)$ denotes the signature signed on document D by host MSMA using its secret key S_{MSMA} and *t* is the timestamp at which the signature is generated. *t* is unique for all routes within a dispatch tree. The addresses of PH and CH only appear in the signature for verification.

Starting the binary dispatch process with secure routes, the agent A_{MSMA} dispatches two PWAs to different hosts, each being encapsulated with an encrypted route for future dispatch task. We call them the first left PWA (PWA_{1L}) and the first right PWA (PWA_{1R}). When an agent has successfully arrived at the current host CH, the carried route r(CH) can be decrypted with the secret key of CH so that the agent can know:

- it is a PWA or a WA. This is used to determine the next task of the agent;
- the signature signed at host MSMA $S_{MSMA}(PWA, ip(PH), ip(CH), ip(RH), r_L, r_R, t)$ for a PWA, or $S_{MSMA}(WA, ip(PH), ip(CH), ip(MSMA), t)]$ for a WA.

If it is a PWA, it will also know

- the address ip(RH) of the right child host RH;
- the encrypted route r_R for the right child agent, which can only be decrypted by the right child host;
- the encrypted route r_L for the left dispatch.

If it is a WA, it will know the address of MSMA, ip(MSMA), the home host where A_{MSMA} is residing. With this address, the WA can send its result to $A_{MSMS.}$

Clearly, under this model, at any layer, only the address of the right child agent is exposed to the current host so that the right dispatch can be completed. For a PWA, if it has $m=2^k$ members, only *k* addresses of its members are exposed to the host.

2.3 Algorithm for Agent Dispatch with Secure Routes

The algorithm for dispatching agents is described as follows:

Algorithm 1: Binary dispatch with secure routes

Step 1: when an agent A is successfully dispatched to host CH, it will use the secret key of CH, S_{CH}, to decrypt the carried route r(CH).

$$r=S_{CH}[r(CH)]$$

Step 2: if A is a WA, go to step 6, otherwise, A is a PWA, it will dispatch another agent to ip(RH), encapsulating the route r_R to it.

Step 3: if the dispatch is successful, host RH will send a message including its signature to CH.

$$msg1=S_{RH}(Entity_{RS}, ip(RH), t)$$

where $Entity_{RS}$ is the full entity of the dispatched agent including its code, state and data. *t* is the timestamp when the agent is received successfully.
Once getting such a message, host CH will keep $S_{RH}(Entity_{RS}, ip(RH), t)$ in its database as a successful dispatch record.

Step 4: Now A should try to complete its left dispatch. Let $r=S_{CH}[r_L]$

Step 5: if A is still a PWA, go to step 2, otherwise go to step 6

Step 6: A starts its task for data accessing

Step 7: when the data-accessing task is completed, A will dispose after successfully sending a message to agent $A_{MSMA,}$

$$msg2=P_{MSMA}[ip(PH), ip(CH), Result_{CH}, S_{MSMA}(WA, ip(PH), ip(CH), ip(MSMA), t_1), S_{CH}(ip(PH), ip(CH), Result_{CH}, t_2)]$$

where $S_{MSMA}(WA, ip(PH), ip(CH), ip(MSMA), t_1)$ is the signature from MSMA, which is included in the decrypted route of the agent. Here it is used for showing the identification of the agent. $S_{CH}(ip(RH), ip(CH), Result_{CH}, t_2)]$ is the signature generated by current host CH. $Result_{CH}$ is the result obtained at CH. PH is the parent host of CH and $t_2>t_1$.

3 Resolving Security Threats

In this section, we will examine several security issues that will be encountered when dispatching mobile agents and show how our model resolves them.

3.1 Preventing a PWA from Dispatching a Child Agent

During the period of dispatching a child agent, a malicious host may peek the code of the agent and make it skip the dispatch process in certain layer after the route is decrypted. Note that skipping a host would mean skipping all other addresses that may be triggered by that host. In the worst case, assuming host H_1 is the malicious one, as shown in Figure 1, if the dispatch of A_5 from H_1 is not in fact performed, those agents in the group including A_5 to A_8 will not be activated. This means the successful interception to the dispatch of a PWA will affect all members included in the aborted PWA. However this attack can be detected in this model.

Taking the case in Figure 1 as an example, if H_1 makes A_1 skip the process of dispatching agent A_5, agent A_{MSMA} cannot receive any messages from each agent of A_5, A_6, A_7 or A_8. If this happens, since the four agents belong to the same group led by agent A_5, A_{MSMA} will suspect first that A_5 may have not been dispatched. A_{MSMA} will ask hosts H_1 and H_5 to show whether the predefined dispatch has been performed. Apparently, if the dispatch has been carried out, H_1 will receive the confirmation message with the signature $S_{H5}(Entity_{A5}, ip(H_5), t)$ from H_5. H_1 cannot forge this signature without H_5's secret key. So, no matter what H_1 claims, the attack can be detected.

If the skipped dispatch is for a WA, such as A_7 doesn't dispatch A_8, it can also be detected since H_7 cannot show a correct signature from H_8 to show the dispatch is successful.

3.2 Route Skip Attack

There is yet another case that can be handled in this model. Consider a partial dispatch route: PWA A_i at host H_i dispatches A_j to H_j and A_j dispatches A_k to H_k, or there are more PWAs between A_i and A_k. In this model, the encrypted route encapsulated to a PWA includes the encrypted route for its right child agent, which can only be decrypted at the child's host in the dispatch route. That means when a PWA is dispatching an agent, it does not know what the agent is, a PWA or a WA, and how many members the agent has. So the case described above that A_i directly dispatches A_k is not likely to take place without the involvement of A_j. That is why the encrypted route is in a nested structure. In the worst case, even if H_i can successfully predict that H_k is its descendent in the dispatch route and makes A_i dispatch a forged agent to H_k, the attack will not be successful either.

Suppose A_k is a WA, the forged route for A_k should be

$$r(H_k)'=P_{Hk}[WA, ip(H_i), S_{MSMA}(WA, ip(H_i), ip(H_k), t)],$$

while the genuine route should be

$$r(H_k)=P_{Hk}[WA, ip(H_j), S_{MSMA}(WA, ip(H_j), ip(H_k), t)]$$

The genuine $r(H_k)$ can only be obtained at H_j when A_j arrives there and decrypts its route. So if A_i want to forge A_j, it must be able to forge $S_{MSMA}(WA, ip(H_i), ip(H_k), t)$. Otherwise, the attack will be detected if the address of parent host in the signature is not $ip(H_i)$. Furthermore, the signature is also required to be included in the returned result for the verification by A_{MSMA}. So since forging the signature is impossible, this kind of attack cannot success.

3.3 Tampering a PWA to Dispatch an Agent to a Wrong Host

Since the hosts are in a competitive situation, if a malicious host knows a host where an agent will be dispatched from it, and the remote host may probably offer a better service than itself, it may tamper the address so that the agent can be dispatched to another host which is known not to be able to provide a competitive offer. The tamper can be done just after the encrypted route is decrypted. However, when an agent is dispatched to a wrong host, its encrypted route will not be correctly decrypted there. Without the correct route, the verification process cannot be undertaken. Even if the destination host can get the correctly decrypted route, the route will show that is a wrong destination since the address of the destination host is included in the signature in the route generated by MSMA that cannot be tampered with. Thus, in both situations, the attack can be detected by the destination host and the agent will be returned to the sender. Meanwhile, this error will be recorded by the destination host for future investigation.

3.4 Sending the Result of a WA to A_{MSMA} Directly or Not

In this model, when a WA has fulfilled its data-accessing task, it will send a message to A_{MSMA} directly by encrypting the result, the signature by the host as well as the signature by the MSMA originally included in the agent's route. The structure is shown as message (2) in section 2.3. The whole message is encrypted with the public key of MSMA so that it can only be decrypted by agent A_{MSMA}. We choose this way in

this model with regard to both security and performance issues. An alternative is that a PWA should be responsible for dispatching agents and collecting data from them. If PWA A_i dispatched PWA A_j which dispatched WA A_k and A_k encrypted its result with the public key of MSMA and sent it to A_j where H_j cannot decrypt. To send the whole result set to A_i, A_j should encrypt its own result together with the encrypted result from A_k. If they are put as two separate encrypted results, deletion or tamper attacks may easily occur in the returning path especially when a large number of results are sent to a PWA. Meanwhile, this will increase the burden of a PWA and the performance will definitely become worse.

A possible solution preventing the results from being tampered or deleted that may take place at any host where a PWA resides is for the receiving side to send a reply to the sending side, just like the process for dispatching. The reply should be a signature generated on the received message by the secret key of the receiving side. In this way, deletion and tampering can be detected by the verification among the MSMA, sending side and receiving side. However, the performance will become inferior.

In comparison, in our model, since a WA only visit one host, the host would not delete the result or prevent its offer from being returned once the agent has been successfully dispatched there. In case the attack occurs, based on the detection of successful dispatch, the problem should be with the side of the host where the agent has arrived. In terms of performance, since each WA has different starting time and ending time for the data-accessing task and each offer will be in small size, the returned results can hardly cause the A_{MSMA} to become a bottleneck.

3.5 Replay Attack

In a malicious host, the replay attack may occur. Consider the following scenario, that a malicious H_i who has a PWA residing in it and it dispatched agent A_j to host H_j. After the normal process has been completed, H_i may replay the dispatch with a forged agent so that on one hand it can get the offer information from H_j constantly and periodically if H_i tampers the agent so that it sends the result to H_i, and on the other hand, excessive agents may jam H_j. However, when an agent is dispatched from H_i to H_j as a replay attack, the timestamp included in the signature from MSMA cannot be tampered with. By verifying the signature, H_j can easily detect the replay attack and H_i will face the risk to be reported.

Similarly, another type of replay attack is for a host, which a WA had earlier resided, to repeatedly counterfeit the WA and send messages to the agent A_{MSMA}. Since the A_{MSMA} is the root agent, it will be disposed of once all WAs have completed their tasks successfully. In addition, if A_{MSMA} repeatedly receives offers from the same host, it will close the communication channel and start an investigation.

3.6 Collusion Attack

If in a normal sequence, host H_a should dispatch an agent to H_b. Assuming H_a and H_c are in a collusion tie, the agent is dispatched to H_c. In this way H_a and H_c make an attempt to skip the visit to H_b who is their competitor and send their own offers instead. However H_c can hardly forge the signature by H_b that should be included in the message returned to A_{MSMA}. In such a case, the counterfeited message can be

detected when it is returned and this will cause the investigation against H_c and H_a. Since H_b will report that no such agent has ever been dispatched to it and H_a cannot show the correct dispatch record which should include the signature by H_b, the attack can be identified. The attack can be successful only when H_a, H_b and H_c make a collusion attack sending a result from H_b encapsulating the price from H_c. However, in a healthy competitive environment, the probability is fairly low. Even if it can take place, the future negotiation or buying agents will visit H_b not H_c and if H_b cannot offer the goods with the provided price, it will result in a commercial cheating, which is the same as a merchant's giving a nominal price and causing the abortion of the purchase. This will cause the deduction of the merchant's credit standing and little agents will be dispatched later to such merchants.

4 Robustness Enhanced Extension

So far we have presented a security enhanced dispatch model for mobile agents. However, like Westhoff's model [13], each PWA only knows the RH to which its right child agent should be dispatched at a certain stage and should the host where the right child agent should go be unavailable, the right dispatch branch cannot be deployed and all the members grouped in this agent will thereby not be activated.

As mentioned in the section 2.1, the binary dispatch model is robust in that a PWA can know all the destination addresses of its children agents. It can choose any of them to be the right child PWA. However, its robustness is built on the basis that all these addresses are exposed to the host. Therefore, its robustness is not feasible with regard to the security. Anyway, it is clear that a PWA should have an alternative for dispatching its right child agent so that if the predefined right child agent cannot be successfully dispatched due to some reasons from the destination host, the PWA can have another route for the right dispatch.

Li proposed a robust model in [14] for serial migration of agents and the route robustness is enhanced by dividing a route, say {ip(H_1), ip(H_2), …, ip(H_n)}, into two parts, say {ip(H_1), , …, ip(H_i)} and {ip(H_{i+1}), …, ip(H_n)}, which are distributed to two agents A_1 and A_2 respectively. A_1 and A_2 are in partner relationship. Each agent residing at any host knows the addresses of the next destination and an alternative host. The latter is encrypted by the public key of its partner agent. In case the migration cannot be performed, the encrypted address will be sent to the partner agent for decrypting. With its assistance, the agent can continue its migration.

The problem for Li's model is that since A_1 and A_2 are two agents that should dynamically migrate, when one needs the other's assistance, locating each other will be costly for both time and system resources though some mechanisms have been proposed by [15], [16]. Meanwhile, the model is a serial one so it is not efficient. Additionally, using the secret key of a dynamically migrating agent is not secure. But the idea of using the mutual assistance of the two agents to enhance the robustness is good and can be easily used in our model, where the two first PWAs in the left and right branches can do it better. Since they don't need to migrate, sending messages to them is fairly simple and fast. Encrypting and decrypting the route using the keys of the host where the first PWA resides is more secure.

For robustness, the route structure in equation (1) can be extended as follows:

(1) For a PWA at CH, r(CH)=P [PWA, ip(RH), r , r , r ', S (PWA, ip(PH), ip(CH), ip(RH), r_L, r_R, r_R', t)], where r_R'=P_{APWA}[ip(SH), r(SH), S (ip(SH), r(SH), t)] is the substitute route for the right branch of host CH, SH is the new substitute host.
(2) For a WA at CH, r(CH)=P [WA, ip(PH), ip(MSMA), S (WA, ip(PH), ip(CH), ip(MSMA), t)]

(2)

In route structure (2), r_R' is encrypted by the public key of the first PWA in another branch of the whole dispatch tree, which here is termed as Assistant PWA (APWA).

Suppose A_1 is the first PWA in the left dispatch sub-tree. A_m is the right one. If current host CH is the descendent of A_1, then r_R' is encrypted by the public key of A_m, P_{Am}. Otherwise, if CH is in the right dispatch sub-tree from the root node, r_R' is encrypted by P_{A1}.

If the dispatch failure occurred when A_i is dispatching A_j, and A_i is in the left dispatch sub-tee, A_i should report it to A_m attaching the substitute route r_R'

$$msg1=P_{Hm}[ip(H_j), ip(H_i), r_R', S_{Hi}(ip(H_j), ip(H_i), r_R', t)]$$

When A_m gets such a message, it will

Step 1: Detect whether H_j has got down. If it is true, then go to step 2, otherwise go to step 3

Step 2: Am will decrypt r_R', r=S_{Hm}[r_R'], and send it to A_i through a message

$$msg2=P_{Hi}[ip(SH), r(SH), S_{MSMA}(ip(SH), r(SH), t_1), S_{Hm}(ip(SH), r(SH), S_{MSMA}(ip(SH), r(SH), t_1), t_2)]$$

Stop.

Step 3: If A_j is in the correct state, A_m will tell A_i about it and record the request in a database.

There are two reasons for A_i to send a request to A_m. One is that H_j has a temporary failure when A_i is trying to dispatch an agent there. Another reason is that host H_i is malicious and attempts to know more addresses by sending a cheating request. However, the failure report will be confirmed by A_m before replying any decrypted routes. And the request is saved by A_m for future investigation.

In this way by route structure (2), a PWA will have a substitute route for the dispatch of its right child agent. Once the original dispatch is not successful, with the assistance of its APWA, it can have another destination to dispatch.

What we should address is that the substitute host is originally included in the members for the right dispatch branch. Taking the dispatch tree in Figure 1 as an example, if the dispatch failure occurred when A_1 is dispatching A_5, A_1 can get an substitute route with the assistance of A_9. Suppose the substitute host is H_6, A_1 will dispatch an agent A_6 to H_6 and A_6 will deploy the right dispatch branch. To be more fault-tolerant, the address of H_5 will still be included in this branch. But it is put to be a leaf node so that A_5 will become a WA only for another attempt to dispatch it. Suppose the new sequence is A_6, A_7, A_8 and A_5, in which A_8 will make another attempt to dispatch A_5. If the dispatch problem with A_5 is temporary, a later attempt will be successful so that in such a case, all hosts will be visited as usual. If the dispatch failure occurred again, the reply from A_9 will show that A_5 is a WA and no more substitute route will be provided.

5 Discussions and Conclusion

In the proposed model, we aim to expose only the necessary addresses to hosts. If a PWA A_i has 2^k agents in its whole branch, only k addresses are exposed to host H_i since these agents should be dispatched directly by A_i in different layers. As a matter of fact, a PWA does not know what the dispatched agent is, how many members it has and with the security mechanisms attacks can be detected. Since this model adopts parallel dispatch, the dispatch efficiency is high.

As Westhoff's model [13] adopted a fully serial migration, the migration complexity is $O(n)$ if there are n hosts to be visited and it provides secure route structure without any robustness mechanism. Li's model [14] ensures both security and robustness. As the addresses of n hosts are distributed to two agents, the whole migration time can be theoretically half of that of the first model. However the time complexity is $O(n)$. In comparison, in our model the efficiency is greatly improved while both the security and robustness are ensured. Either the fully binary dispatch model or the model with 1 substitute route, the dispatch complexity is $O(\log n)$.

With regard to the complexity for generating routes, three models have different performances. As pointed by [13], when the route adopts the nested structure, it will help to prevent route tampering or deleting attacks and detect them as early as possible. The nested route structure is also adopted by Li's model and our model. Based on this condition, taking the time for encrypting a route as a constant for simplifying, the complexity for generating routes can be estimated as follows.

For Westhoff's model, the route with n addresses can be generated after the route with $n-1$ addresses has been generated. So, the complexity $T(n)$ can be calculated as $T(n)=O(n)$ from the following,

$$\begin{cases} T(n)=T(n-1)+C \\ T(1)=C,\ C \text{ is a constant} \end{cases}$$

For Li's model, suppose the hosts in the predefined sequence are $\{H_i, H_{i-1}, H_{i-2}, H_{i-3}, \ldots, H_2, H_1\}$, if host H_{i-1} is not reachable, H_{i-2} will become next destination from H_i and H_{i-1} will never be visited for this journey. So the generated normal route with $i-3$ addresses will be used for generating the substitute route with $i-2$ addresses. The route generating complexity with 1 substitute route is

$$\begin{cases} T(n)=T(n-1)+2C \\ T(1)=C \end{cases}$$

And $T(n)$ is $O(n)$.

In our model, the complexity for generating routes without substitute routes is $T(n)=O(n)$, where T(n) is

$$\begin{cases} T(n)=2T(n/2) \quad (n=2^k) \\ T(i)=2T(i/2)+C \quad (2 \le i \le 2^{k-1}) \\ T(1)=C \end{cases}$$

When generating the first substitute route for a branch, only a few steps should be taken in the left sub-branch of this branch. The number of the steps is up to the height h of the sub-branch. The complexity for the our model generating 1 substitute route is

$$\begin{cases} T(n)=2T(n/2)+C \quad (n=2^k) \\ T(i) \le 2T(i/2)+hC \quad (h \le k-1,\ 2 \le i \le 2^{k-1}) \\ T(1)=C \end{cases}$$

And hereby $T(n)$ is $O(nlogn)$.

In our model, a failed host will be tried for a second time while Li's model skips it. Otherwise the complexity of Li's model for generating routes will become extremely worse since the sequence of hosts in the substitute route has been changed and the route should be generated again. When a route includes 1 substitute route, the complexity will be $T(n)=T(n-1)+T(n-2)+2C$, $T(1)=C$ and $T(n)$ is $O(2^n)$.

For future work, we will work toward a global e-commerce framework with security mechanisms that is suitable for parallel processing by mobile agents. Some improvements should be done to current model to provide more substitute route with less loss of time complexity and the evaluation model on both security and commercial credit is also needed since in our model the hosts where APWAs reside are the most important to global dispatch. Based on this environment, activities on merchant assessment, information gathering and negotiation can be deployed by mobile agents automatically and safely.

Acknowledgement. This work is partially supported by the research grant of Strategic Program on Computer Security from NSTB of Singapore.

References

1. Sohn, S. and Yoo, K. J.: An Architecture of Electronic Market Applying Mobile Agent technology. Proceeding of 3rd IEEE Symposium on Computers and Communications (ISCC '98), Athens, Greece, (1998) 359-364
2. Corradi, A., Montanari R., and Stefanelli C.: Mobile Agents in E-commerce Applications. Proceedings of 19th IEEE International Conference on Distributed Computing Systems, Workshops on Electronic Commerce and Web-based Applications, Austin, Texas, USA, (1999) 59-64
3. Chrysanthis, P., Znati, T., Banerjee, S., and Chang, S.K.: Establishing Virtual Enterprises by means of Mobile Agents. Proceeding of Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia, (1999) 116-123
4. Rodrigo, T. D. and Stanski, A.: The Evolving Future of Agent-based Electronic Commerce. In Rahman S. M. and Raisinghani M. S.Electronic (eds.): Commerce: Opportunity and Challenges, Idea Group Publishing, Hershey, USA, (2000) 337-351
5. Silva, L., Batista, M., Martins V., and Soares, G.: Using Mobile Agents for Parallel Processing. Proceeding of International Symposium on Distributed Objects and Applications (DOA'99), Edinburgh, Scotland, (1999) 34-42
6. Papastavrou, S., Samaras G., and Pitoura, E.: Mobile agents for World Wide Web distributed database access. IEEE Transactions on Knowledge and Data Engineering, Vol. 12, Issue 5 , (2000) 802 –820
7. Karjoth, G., Lange D.B., and Oshima, M.: A Security Model for Aglets. IEEE Internet Computing, July-August (1997) 68-77
8. Varadharajan, V.: Security enhanced mobile agents. Proceedings of the 7th ACM conference on Computer and communications security, Athens, Greece, (2000) 200 – 209
9. Ubayashi, N., Tamai, T.: RoleEP: role based evolutionary programming for cooperative mobile agent applications. Proceedings of International Symposium on Principles of Software Evolution, (2000) 232 –240

10. Wayner, P.: Digital Copyright Protection. SP Professional, Boston, USA, (1997)
11. CCITT Recommendation X. 509-1989: The Directory-Authentication Framework. Consultation Committee, International Telephone and Telegraph, International Telecommunication Union, Geneva (1989)
12. Lange, D., and Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley Press, Massachusetts, USA (1998)
13. Westhoff, D., Schneider, M., Unger, C. and Kenderali, F.: Methods for Protecting a Mobile Agent's Route. Proceedings of the Second International Information Security Workshop (ISW'99), Lecture Notes in Computer Science, LNCS 1729, Springer-Verlag, Berlin Heidelberg New York, (1999) 57-71
14. Li, T., Seng, C.K. and Lam, K.Y.: A Secure Route Structure for Information Gathering. Proceedings of 2000 Pacific Rim International Conference on AI, Melbourne, Australia, (2000)
15. Belle, W.V., Verelst, K. and D'Hondt, T.: Location transparent routing in mobile agent systems merging name lookups with routing. Proceedings of 7th IEEE Workshop on Future Trends of Distributed Computing Systems, (1999) 207 –212
16. Maass, H.: Location-aware Mobile Applications Based Directory Services. Mobile Networks and Applications, Baltzer Science Publishers BV, (1998)