Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Standards & Interfaces 28 (2006) 600–611

COMPUTER STANDARDS & INTERFACES

www.elsevier.com/locate/csi

Protecting mobile-agent data collection against blocking attacks

Min-Hua Shao[a,*], Jianying Zhou[b]

[a]*Institute of Information Management, National Chiao Tung University, 9F., No.433, Fude St., 1001 Ta Hsueh Road, Hsinchu 300, Nangang District, Taipei 115, Taiwan, ROC*
[b]*Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore*

Received 15 October 2004; received in revised form 25 January 2005; accepted 11 February 2005
Available online 19 March 2005

Abstract

Full-scale adoption of mobile agent technology in untrustworthy network environment, such as Internet, has been delayed due to several security complexities. The protection of mobile agents against the attacks of malicious hosts is considered a very challenging security problem. It has inspired lot of research interest, but very few measures exist to counter *blocking attack* where a host with malicious intentions refuses to transmit a mobile agent to the next host. It becomes an important requirement for the agent owner to rescue the data collected by the agent under custody and redeem a loss. In this paper, we present two schemes that rescue the offering results from a malicious host's blocking attack, and make a comparison of their performance from several aspects. Our approach has two new features that previous protocols lack. It allows the proper handling of time-sensitive offers and supports the gradual decision-making execution.

Keywords: Mobile agent; Fault tolerance; Blocking attack; Time-sensitive offers; Gradual decision-making execution

1. Introduction

A *mobile agent* is a program that represents a user in computer networks and can migrate autonomously from node to node, to perform some computation on behalf of the user. A variety of applications for mobile agent technology have been introduced into electronic commerce such as on-line shopping, information retrieval, etc. The introduction of mobile code into a network raises several security issues [7], in which protection of mobile agents from malicious hosts is considered one of the most challenging security problems in mobile agent systems [4,5,8].

A malicious host could try to get some profit from a mobile agent by reading or modifying the code, the data, the communications or even the results due to its complete control on the execution. Bierman and Cloete classified the security threats that mobile agents can possibly encounter from their executing hosts [2]:

- *Integrity Attacks*—integrity interference and information modification.

* Corresponding author. Tel.: +886 656 874 8272; fax: +886 656 775 5014.
E-mail address: mhshao@alumni.nccu.edu.tw (M.-H. Shao).

0920-5489/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2005.02.002

- *Confidentiality Attacks*—eavesdropping, theft, and reverse engineering.
- *Authentication Risks*—masquerading and cloning.
- *Availability Refusal*—denial of service, delay of service, and transmission refusal.

Most of the available countermeasures have focused on integrity and confidentiality attacks, and very few measures exist to counter availability refusals and, especially, *transmission refusal*. When a host with malicious intentions refuses to transmit the agent to the next host, either on a predetermined path or determined by the agent based on dynamically gathered information, *blocking attack* occurs. Then, the advantage on the use of the vast amount of resources available on the Internet may be lost or severely obstructed. Therefore, how to protect a mobile agent against blocking attack and rescue the data collected by the agent under custody is an important requirement for the agent owner to redeem a loss. This is actually an issue related to *fault tolerance*, which is crucial to enable a reliable environment. This paper will be focused on the solutions of rescuing the agent data from malicious hosts' blocking attacks.

Besides protecting mobile agents against blocking attacks, our approach also allows the proper handling of time-sensitive offers and supports gradual decision-making execution—the features that previous protocols lack. If a mobile agent is used to collect time-sensitive offers, i.e., offers that lose value over time, the mobile agent system may need to enable the agent owner to duly receive partial results during its journey. In addition, how to strengthen the relevance between the offers collected by the agent and its owner's needs should be considered. An agent owner usually wants to communicate with or control the agent when it travels on its itinerary. Gradual decision-making execution is essential to this purpose, which means the agent owner can timely modulate some preferences during its execution.

The rest of this paper is structured as follows. In Section 2, we outline the properties of fault tolerance on mobile agents, and further survey related work and point out their weaknesses. After that, we propose a new approach to protect against blocking attack in Section 3, and give a comparison of different scenarios on the application of our approach in Section 4. In Section 5, we further extend our approach for blocking attack protection to time-sensitive offers and gradual decision-making model. Finally, we conclude the paper in Section 6.

2. Fault tolerance

Fault tolerance is crucial to enable massive use of mobile agent technology in today's business applications. The mechanisms of fault-tolerance have been used in the three main parts of a mobile agent system: Agent manager, platform agencies and the mobile agents [16]. Two important properties of fault tolerance for the execution of mobile agents are *exactly-once* and *non-blocking* [12]. The exactly-once property given by Rothermel and Strasser [13] is as follows. Exactly-once used for the failure semantics of a single remote procedure has been already defined for *remote procedure call* (RPC) systems. In the context of mobile agents, a sequence of agent stages is to be considered rather than a single procedure. An agent execution is defined to be "exactly once" if the entire sequence of its stages is eventually performed, and all operations of each stage are executed exactly once. Therefore, the exactly-once property for mobile agents can be achieved in a simple way by using transactional message queues [17]. Message queues can deal properly with asynchronous communication between processes dwelling on the same or different nodes in order to provide for persistent messages and ensure the exactly-once delivery.

In terms of non-blocking, the failure of a component (i.e., agent, host and especially malicious host, or communication link) can result in blocking during the mobile agent execution. In academic literature [1,6,14,15], *replication* is a widely adopted mechanism to prevent blocking except for malicious host. The illustration of mobile agent replication is shown in Fig. 1. In order to support agent fault-tolerance, copies of the agent are additionally sent to a non-empty set of other places that provide the logical execution environment for the agent. Obviously, it may lead to multiple executions of the agent and divert from the exactly-once property. There have been some discussions of giving consideration to two properties simultaneously [12,16]. However, much of that work only focused on crash failures but not

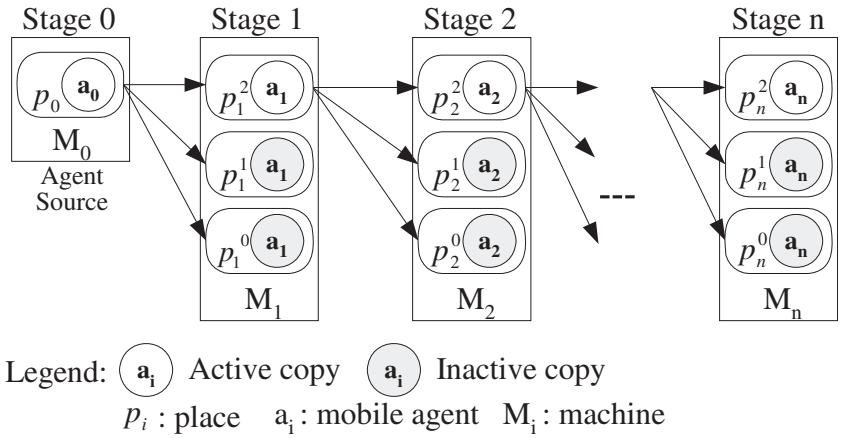


Fig. 1. The general execution model of mobile agent replication.

malicious failures. So, in terms of the failure of a component including agent, machine, or communication link, they're quite workable, but not in terms of transmission refusal. After a free-roaming mobile agent visits a host, it might be blocked for leaving the malicious host. Then, the agent will not be able to return to its originator, and the originator cannot get the data collected by the agent. Therefore, the detailed treatment of dealing with blocking attacks from the malicious host is a focus of our current work. That is, how to make a mobile agent and its collected offers return to the originator reliably even if there are malicious hosts visited by the agent is considered in this paper.

3. Our schemes against blocking attack

Here we propose a new approach to protect a mobile agent against blocking attack, in which the agent owner can detect custody of its agent and simultaneously retrieve the offering results from the agent under suspicious blocking.

3.1. Assumptions and notation

Because agents might traverse untrusted hosts or networks, users will have much lower levels of mutual trust. The security-related requirements fall into the following categories: agent privacy and integrity, agent and server authentication, authorization and access control, metering, charging, and payment mechanisms [7]. Public-key cryptography is used widely for the establishment of authenticated communication channels. For online commercial services, most merchants need to provide their certificates to customers for service authentication. On the other hand, it is unsuitable to carry secret or private keys with agents for authentication purposes, because this leaves them vulnerable to malicious hosts. As for data confidentiality, secret-key cryptography can be employed in consideration of efficiency. It is also used for message sealing or message digests to detect any tampering of the code or data. These cryptographic primitives and security mechanisms have been applied in some of existent mobile-agent systems. Telescript is the first system designed expressly to support mobile agents in commercial applications, where agent transfer is authenticated using RSA and encrypted using RC4 [18]. Ajanta is the Java-based mobile-agent system in which transfer is encrypted using DES and authenticated using ElGamal protocol [9]. A public key infrastructure (PKI) is assumed in the mobile agent environment. Revocation of certificates is the key to the operation of PKI, and certification revocation list (CRL) posted in network-wide directories is a popular mechanism being adopted, although there are a large variety of related work such as Certificate revocation tree (CRT) and so on [10,11,19].

Each host S_i has a certified private/public key pair $(\bar{v}_i, v_i)$. Given a signature expressed as $Sig_{\bar{v}i}(m)$, we

assume that anyone could deduce the identity of S_i from it. The chain of encapsulated offers O_1, O_2,..., O_n is an ordered sequence. Each entry of the chain depends on some of the previous and/or succeeding members. A chaining relation specifies the dependency. An important definition of an agent given by Cheng and Wei [3] is as follows. An agent is defined as $A=(I, C, S)$ where I is the identity, C is the code and S is the state of the agent. Both I and C are assumed to be static while S is variable. I is in the form of (ID_A, Seq_A), where ID_A is a fixed identity bit string of the agent and Seq_A is a sequence number which is unique for each agent execution. The originator S_0 signs h_A, where $h_A=H(I, C)$ is the agent integrity checksum and $Sig_{\bar{v}o}(h_A)$ is the certified agent integrity checksum. The agent carries this certified checksum, allowing the public to verify the integrity of I and C and deduce the identity of S_0.

This paper will be focused on the solutions against blocking attack. A thorough discussion about a mobile-agent execution is beyond the scope of this paper, and the reader is referred to [20]. The notation used in the scheme description is summarized in Table 1.

3.2. Scenario

Consider a shopping scenario in which a mobile agent denoted as MA_A departing from host S_0 will obtain a list of offers $(O_1, O_2,..., O_n)$ from different hosts S_1, S_2,..., S_n selected dynamically when the agent roams over the network. It verifies S_0's signature $Sig_{\bar{v}o}(h_A)$ to identify the agent. A simplified model of a mobile-agent execution is shown in Fig. 2(a). In this scenario, suppose a hostile host S_j that is executing the shopping agent performs blocking attacks for its own interest. For instance, S_j may impede an agent with a task of requiring quotation of air-ticket prices to collect the latest market information in order to get good competition. After the expiry of the agent, the originator needs to trigger the agent again to fulfill its task. The custody of a mobile agent is illustrated in Fig. 2(b).

Table 1
Notation

$S_0=S_{n+1}$	The originator
$Si, 1\le i\le n$	A host
$Oi, 1\le i\le n$	An encapsulated offer from S_i that is cryptographically protected in which the identity of S_i is explicitly specified
$O_1, O_2,..., O_n$	The chain of encapsulated offers
MA_A	A mobile agent consist of $A=(I, C, S)$ and other items
$\overline{MA}_A$	$\overline{MA}_A$ differs from MA_A in some preferences $\bar{A}=(I, \bar{C}, S)$
$(\bar{v}_i, v_i)$	Certified private and public key pair of S_i
$Sig_{\bar{v}i}(m)$	A signature of S_i on message m with its private key $\bar{v}_i$
$A\rightarrow B:m$	A sends message m to B

In the context of blocking attacks, we propose a simple scheme and a <t, n> fault-tolerant scheme to minimize loss of offers to the originator S_0.

3.3. A simple scheme

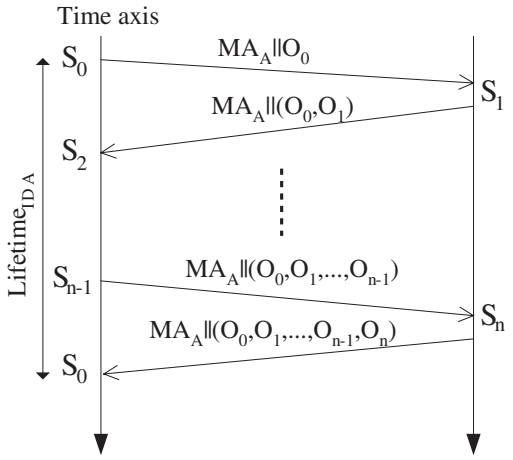
Suppose hosts do not collude. A host S_i may receive an acknowledgement from the next host S_{i+1} to prove that S_{i+1} has sent the agent to S_{i+2}. The acknowledgment $ACK(S_{i+2}, Sig_{\bar{v}i+1},(I, S_{i+2}))$ contains the identities of S_{i+2} and the agent, and is signed by the certified private key of S_{i+1}. A mobile-agent execution with a commitment toward the preceding host is depicted in Fig. 3(a).

If S_i does not receive any feedback from S_{i+1} at the expiry of fault-tolerant time t_f defined in the agent $MA_A=(A, t_f, Sig_{\bar{v}o}(h_A, t_f))$, S_i will send the offering results $(O_1, O_2,..., O_i)$ currently collected by the agent back to the originator S_0. The fault-tolerant time t_f is decided according to the time that a host can spend for executing the agent and the transmission time over network, which are discussed in [5].

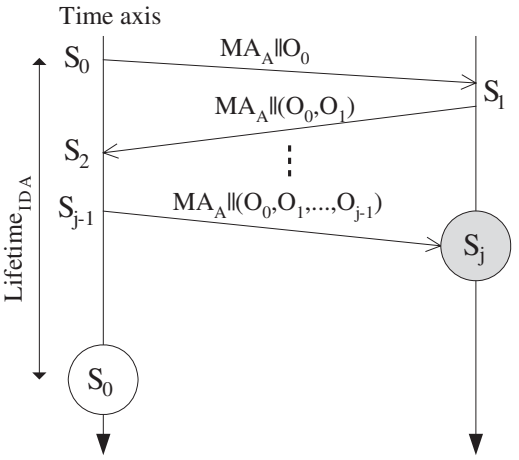
In addition, if the originator S_0 only received the partial offers before the agent's deadline, and the partial offers did not fulfill its needs, S_0 may launch the agent again. In Fig. 3(b), the originator S_0 does not have any loss of the offers (O_1, O_2, O_3) even though a hostile host S_4 refuses to transmit the agent to the next host. However, the identity of S_{i+2} will be disclosed to S_i. This implies a slight weakening of forward privacy. Nevertheless, such a straightforward scheme can provide the agent owner with all offers collected by the agent before being blocked by the misbehaving host.

3.4. A <t, n> fault-tolerant scheme

Obviously, there are heavier communications and computations overheads in the above simple scheme,



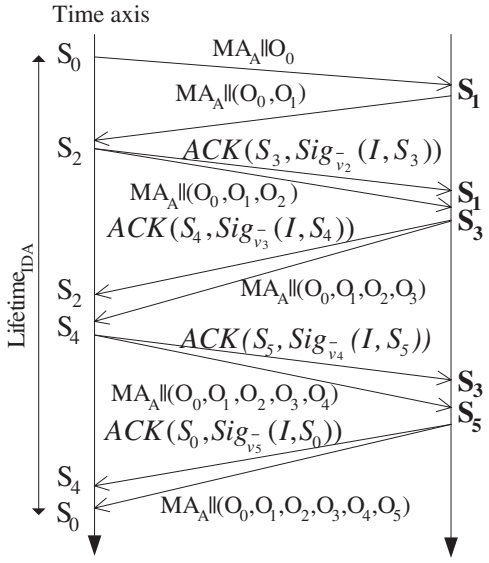
(a) Simplified model of a mobile-agent execution



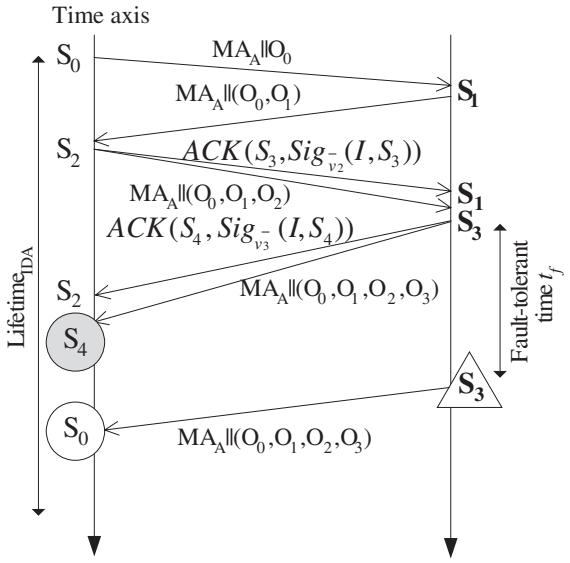
(b) Example of custody of a mobile agent by S_j

Note: $MA_A = (A, Sig_{v_0}(h_A))$

Fig. 2. Illustration of a mobile-agent execution.



(a) A mobile-agent execution with a commitment toward the preceding host



(b) Transmission refusal occurs in S_4

Note: $MA_A = (A, t_f, Sig_{v_0}(h_A, t_f))$

Fig. 3. A simple scheme against blocking attack.

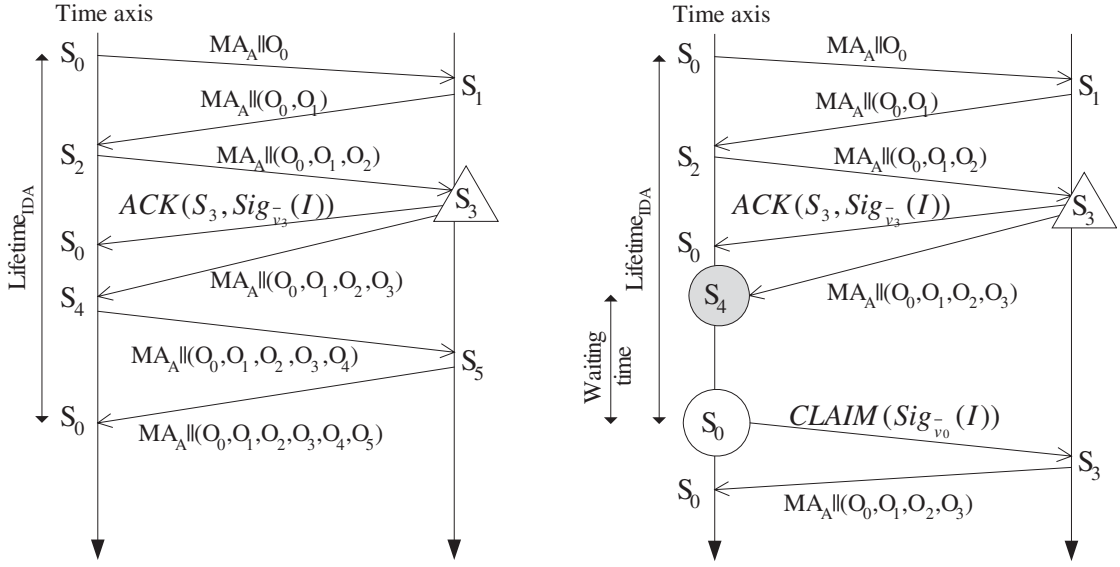
thus the advantage of reducing network uses with the mobile agent technology may be lost.

Here we consider a more efficient solution on detection of blocking attack. The idea of acceptable tolerance with regard to the risk of offering loss for an agent owner is introduced into our new scheme. That is, a fault-tolerant roaming hop n and a fault-tolerant execution time t_f are used to determine the risk of offering loss, and we call it a $<t, n>$ fault-tolerant scheme. Note, the fault-tolerant execution time t_f is used by the agent owner to periodically track the agent's location, which is different from the one used in the simple scheme.

As far as the fault-tolerant roaming hop n is concerned, it will accompany the mobile agent $MA_A=(A, n, Sig_{\bar{v}o}(h_A, n))$ when roaming over the network. Once the number of offers $(O_1, O_2,\ldots, O_i)$ collected by the agent reaches n or a multiple of n, a host S_i where the agent is visiting should send the originator S_0 an acknowledgement, denoted as $ACK(S_i, Sig_{\bar{v}i}(I))$. An example of a mobile agent execution with $n=3$ is illustrated in Fig. 4(a). In case of a blocking attack triggered by S_j, S_0 would not do anything until the agent lifetime expires. By then S_0 will make a claim $CLAIM(Sig_{\bar{v}0}(I))$ for a host Si $(i<j)$ where it gave the latest acknowledgment to obtain the offering results $(O_1, O_2,\ldots, O_i)$ as shown in Fig. 4(b).

Clearly, the originator cannot determine whether the agent has failed, been delayed or blocked. Also, it is a challenge to determine how long a free-roaming agent will take to collect n offers. For example, the execution time of a mobile agent at a host may be determined mostly upon the facilities of the host and its serving load at the moment; and further, the transmission time is mainly hedged about with the network quality. Therefore, the maximum loss of offers depends on the fault-tolerant roaming hop n.

For only using the fault-tolerant roaming hop, the agent owner cannot bring emergency measures into action early in response to the blocking attack occurred; and further, it may suffer a great loss of offers when the attacking event arose at the beginning of a checking point with a big parameter n. Therefore, another parameter of fault-tolerant execution time t_f is



(a) Example of a mobile-agent execution with n=3 and 5 hops

(b) Custody of a mobile agent by S_4 and n=3

Note: $MA_A = (A, n, Sig_{\bar{v}_0}(h_A, n))$

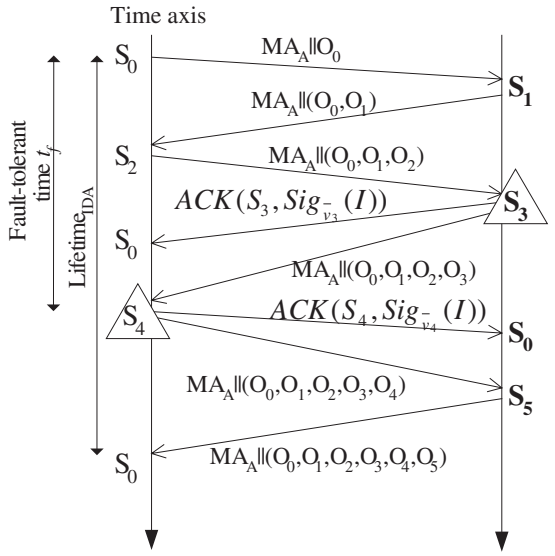
Fig. 4. A fault-tolerant roaming hop approach against blocking attack.

considered. The combination of two fault-tolerant parameters t_f and n along with an agent $MA_A=(A, n, t_s, t_f, Sig_{\bar{v}o}(h_A, n, t_s, t_f))$ (t_s is the agent starting time used with t_f to decide a checking point.) is a better solution with constant overload costs, and also limited loss of the offering results. The agent owner not only passively receives acknowledgements from hosts upon n and t_f but also actively traces the agent's location at the end of every period of t_f.

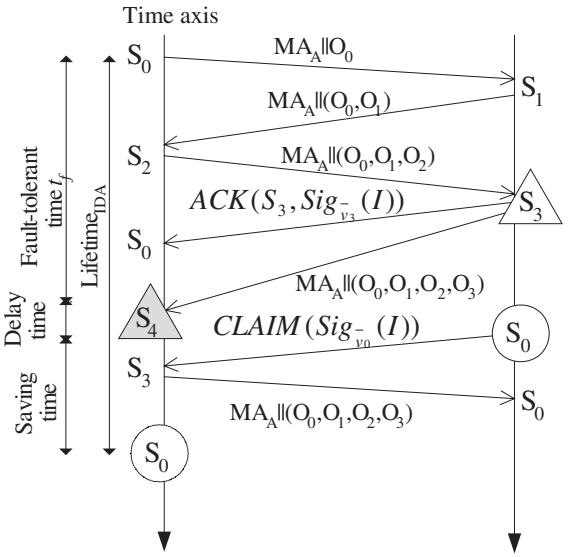
There are two kinds of tracking points in a <t, n> fault-tolerant scheme where the visited host S_i should send an acknowledgement, denoted as $ACK(S_i, Sig_{\bar{v}i}(I))$, to the originator: (i) if the roaming time of a mobile agent has hit on t_f or a multiple of t_f, or (ii) if the number of offers is reaching n or a multiple of n. For instance, in Fig. 5(a) at host S_3 condition (ii) is satisfied with $n=3$, at host S_4 condition (i) is satisfied with $t>t_f$; and further, in Fig. 5(b) the transmission refusal occurs at S_4. In this case, at the end of the first fault-tolerant execution time t_f plus the delay time σ, the originator S_0 will make a request, denoted as $CLAIM(Sig_{\bar{v}0}(I))$, for the offers (O_1, O_2, O_3) from the host S_3 that gave a preceding acknowledgement. Obviously, the agent owner can cope with blocking attacks early in such a way.

If a host visited by a mobile agent is at its checking points defined by n or t_f, this host has to keep the offering data collected by the agent in case the originator wants to retrieve the data because of a blocking attack. In order to lower the storage burden of hosts, there are two ways to release offers. One is that the originator controls the time of releasing offers. S_0 will send a releasing message to the host at the preceding checking point when receiving the following acknowledgement. The other one is that each host on a checking point will set free the offers after the migration of the agent to the next host plus a safe period if it does not receive any claim to offers from S_0. The safe period is often set as a multiple of the fault-tolerant execution time t_f.

In a <t, n> fault-tolerant scheme, a maximum loss of offers is less than n if two kinds of checking points are not placed at the same host; otherwise, there may be n offering losses. How to collocate two kinds of checking points defined by n and t_f is crucial to control the risk of offering loss. There are



(a) Example of a mobile-agent execution with n=3 and 5 hops



(b) Custody of a mobile-agent by S_4 with n=3

Note: $MA_A=(A, n, t_s, t_f, Sig_{\bar{v}0}(h_A, n, t_s, t_f))$

Fig. 5. Illustration of a <t, n> fault-tolerant scheme against blocking attack.

three constructions to present a different level of reliability.

- Strong construction of a <t, 1> scheme is used for a sensitive business transaction over an untrustful network. The fault-tolerant execution time is also set as small as possible. It can let the agent owner to obtain all offering results even though a blocking attack take place; meanwhile, the communication and storage overheads also explode.
- Trusted construction of a <t, ∞> scheme is used for a regular business transaction over a quality-of-service network. The checking points that are set with a loose fault-tolerant execution time alone is sufficient for the agent owner to obtain the offering data collected by the agent. Compared with strong construction, the communication and storage overheads are reduced but some offering results may get lost if a blocking attack occurs.
- Optimized construction of a <t, n> scheme is feasible to build a reliable environment with constant overloading that is larger than trusted construction but smaller than strong construction; and further also limited loss of the offering results that is larger than strong construction but smaller than trusted construction. The two parameters n and t_f could be defined according to the security level of the system. The smaller they are, the securer the system, but the higher the system's overheads.

4. Comparison

In the earlier section, we have presented two schemes and three constructions to show how to alleviate the loss of offering results in case of blocking attacks. Here we compare their performance from several aspects. The comparison result is summarized in Table 2.

We first consider the risk of offering loss. With the simple scheme and strong construction of a <t, 1> scheme, each host in a mobile agent's itinerary always needs to report the agent's status in its migration. On the contrary, the trusted construction of a <t, ∞> scheme is used for a trusted environment where a mobile agent roams freely and fearlessly without being threatened by a possible malicious host. It is not necessary to set the checking points during an agent's roaming journey. As for the optimized construction of a <t, n> scheme, there is an allowable loss of offers upon the setting of two parameters t_f and n. Its maximum loss of offers is n in the worst situation where two kinds of checking points are placed at the same hosts.

It is a trade-off between the risk of offering loss and the communication overheads. Only if the tracking points are set as frequently as possible can the risk of offering loss decrease as small as possible. For the simple scheme and strong construction of a <t, 1> scheme, communication overheads will arise as a result of transferring acknowledgements for each migration of a mobile agent. Considering a trusted execution environment, a trusted construction of a <t, ∞> scheme does not need any additional communications to protect a mobile agent against blocking attacks. The level of communication overheads in an optimized construction of a <t, n> scheme depends on two kinds of checking points. The larger two parameters n and t_f are, the lower communication overheads, and vice versa.

Now we discuss the storage requirements. The storage requirements of maintaining offers in all of visited hosts depend on the frequency of interaction with the agent owner. A higher frequency allows the originator to track more complete trails of a roaming agent. Thus the hosts need to maintain offers for a shorter time. In the simple scheme, the current host needs to prove to the preceding host about the agent's status. With the strong construction, the originator can control the latest track as the agent migrates. In the

Table 2
Comparison of performance

Approach	Simple scheme	<t, n> fault-tolerant scheme		
		Strong construction	Optimized construction	Trusted construction
Risk of offering loss	Minimum	Minimum	Medium	High
Communication overheads	High	High	Medium	Low
Storage requirements	Low	Low	Medium	High
Transparency	Low	Low	Medium	High

optimized construction, the duration of maintaining offers in the hosts of tracking points depends heavily on a multiple of the fault-tolerant execution time t_f. Due to no interaction with the agent owner, all of visited hosts are burdened with a large memory of storage in a trusted construction of a $<t, \infty>$ scheme.

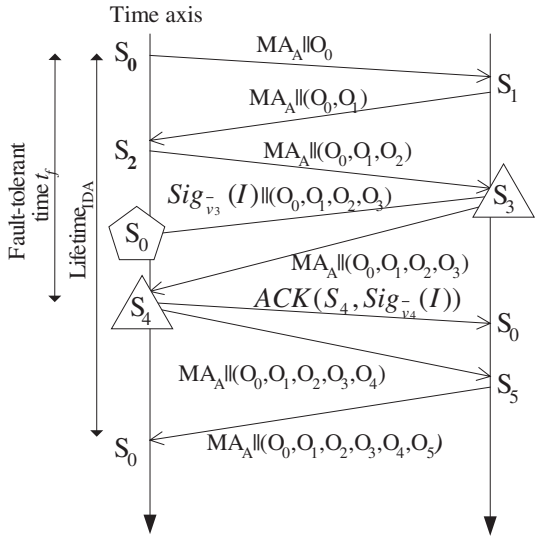
As far as transparency is concerned, an overloading effect of security protections to a mobile agent system should be as small as possible. With the simple scheme and strong construction, each host visited by an agent is involved in the work of protecting an agent from blocking attacks. By contrast, in the trusted construction, hosts in an agent's itinerary are much less involved to deal with extra work for blocking attack protection. In the optimized construction, how many hosts need to deal with the extra work is depended on the parameters of t_f and n.

5. Applications

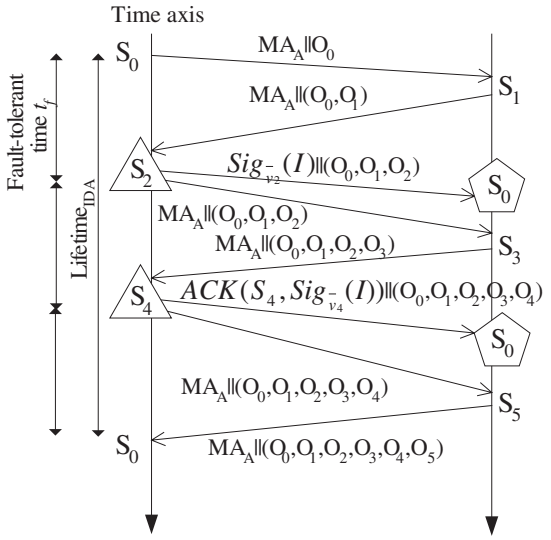
Up to now, information retrieval is a major application of mobile agent technology. Existing schemes in mobile agent systems, however, are too weak for time-sensitive offers that lose value over time such as stock exchange data services. In addition, gradual decision model for information retrieval is often a more efficient way to get the expected data. That is, after a wide-range information retrieval at the beginning, an abstract idea could be converted into concrete demands by narrowing the searching scope. Our approach, a $<t, n>$ fault-tolerant scheme, also allows the proper handling of time-sensitive offers and supports the gradual decision-making execution besides protecting against blocking attacks.

5.1. Time-sensitive offers

Considering time-sensitive offers, the mobile agent system should allow the agent owner to duly receive partial results. The timing of partial offers return can be decided upon the fault-tolerant roaming hop n and the fault-tolerant execution time t_f, respectively. In Fig. 6(a), time-sensitive offers are returned upon the fault-tolerant roaming hop n. Host S_3 on the



(a) Time-sensitive offering return upon fault-tolerant hop n (n=3 and 5 hops)



(b) Time-sensitive offering return upon fault-tolerant time t_f (n=4 and 5 hops)

Note: $MA_A = (A, n, t_s, t_f, Sig_{-v_0}(h_A, n, t_s, t_f))$

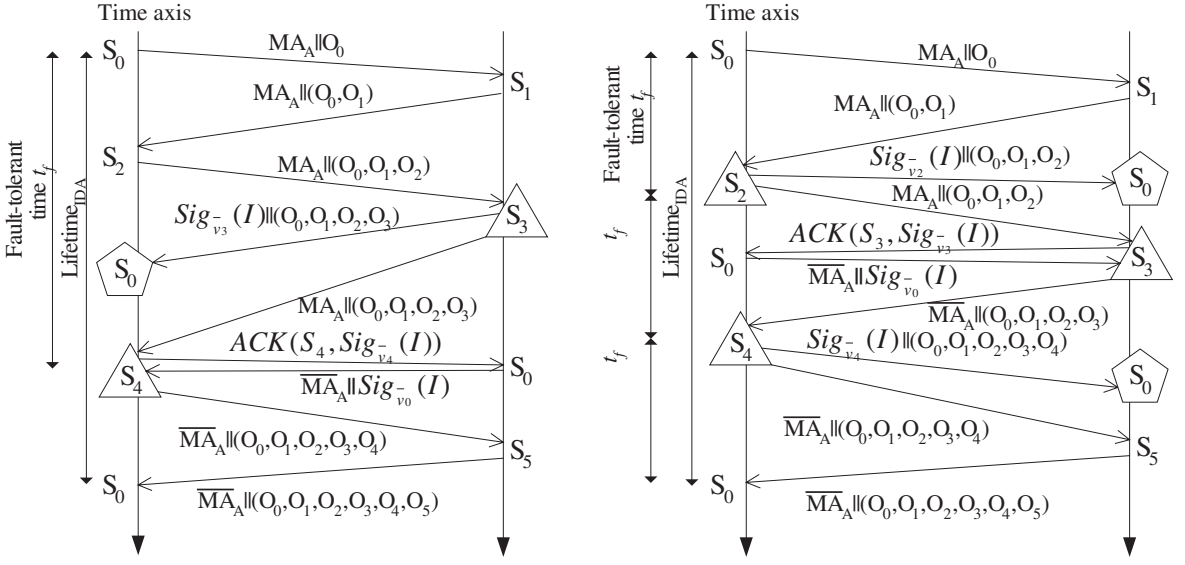
Fig. 6. A mobile agent execution with time-sensitive offers.

checking point $n=3$ should send the partial offers $Sig_{\bar{v}3}(I)||(O_0, O_1, O_2, O_3)$ back to the originator S_0; and further, host S_4 on the checking point t_f will give an acknowledgement $ACK(S_4,Sig_{\bar{v}4}(I))$ to S_0. Another example is in Fig. 6(b), where time-sensitive offers are returned upon the fault-tolerant execution time t_f. Host S_2 placed on the checking point t_f should send the partial offers $Sig\bar{v}_2(I)||(O_0, O_1, O_2)$ back to the originator S_0; and further, host S_4 placed on both checking points t_f and $n=4$ should send $ACK(S_4,Sig_{\bar{v}4}(I)||(O_0, O_1, O_2, O_3, O_4)$ back to S_0.

5.2. Gradual decision-making execution

For information retrieval of an unripe idea, how to strengthen the relevance between the offers collected by a mobile agent and its owner's needs is considered here. Gradual decision-making execution would be helpful to the purpose. A user might need to contact her shopper agent to update some preferences it is carrying. For example, in a scenario that a mobile agent is used to collect offers for some travel package services, a user will sift specific demands, like scenic spots and traveling country, from her rudimentary ideas, such as price and period.

The design of gradual decision-making execution is similar to the handling of time-sensitive offers. Firstly, the timing of sending the partial offers back can be either on the fault-tolerant roaming hop n or on the fault-tolerant execution time t_f. Then, an agent owner can decide whether he wants to update some preferences according to the partial offers. For example, in Fig. 7(a) the originator S_0 receives the partial offers, denoted as $Sig_{\bar{v}3}(I)||(O_0, O_1, O_2, O_3)$, from host S_4 placed on the checking point $n=3$; and further, S_0 receives an acknowledgement, denoted as $ACK(S_4,Sig_{\bar{v}4}(I))$, from host S_4 which is on the checking point t_f as soon as S_0 gives an updated mobile agent, denoted as $M\bar{A}_A||Sig_{\bar{v}0}(I)$ in which $M\bar{A}_A=(\bar{A}, n, t_s, t_f, Sig_{\bar{v}0}(\bar{h}_A, n, t_s, t_f))$ and $\bar{A}=(I, \bar{C}, S)$, to S_4. In addition, an example of updating some preferences early by shortening the fault-tolerant execution time t_f is shown in Fig. 7(b). At the same time, it can also be used for time-sensitive offers.



(a) Gradual decision-making execution with fault-tolerant hop n=3

(b) Gradual decision-making execution with time-sensitive offers (n=3)

Note: $MA_A=(A,n,t_s,t_f,Sig_{\bar{v}_0}(h_A,n,t_s,t_f))$ $\overline{MA}_A=(\overline{A},n,t_s,t_f,Sig_{\bar{v}_0}(\overline{h}_A,n,t_s,t_f))$

Fig. 7. Gradual decision-making in a mobile agent execution.

6. Conclusion

Reliability is essential to business transaction activities. Fault tolerance is one of the reliability mechanisms of considerable importance. The fault-tolerant execution of free roaming agents will appear at the core of mobile agent systems soon. In this paper, a fault-tolerant approach against malicious hosts is discussed. We presented a simple scheme, a $<t, n>$ fault-tolerant scheme and its three constructions to explain how to save the offering results in case of blocking attacks. Moreover, we made a comparison of performance for two schemes and three constructions in order to show their appropriate applications in different occasions.

Besides, we have shown that the $<t, n>$ scheme can be modified slightly, resulting in a scheme which is very well suited for time-sensitive offers and gradual decision-making execution—two new features not discussed in previous mobile agent protocols. In our approach, an agent owner can determine the time to receive the partial offers according to the fault-tolerant roaming hop and the fault-tolerant execution time. Furthermore, a user can timely update some preferences that her shopping agent is carrying when it travels on its itinerary, which is especially well suited for information retrieval of an uncertain or rudimentary idea. Additionally, as a mobile agent often has to communicate or synchronize with each other or the agent owner for the sake of fulfilling its task, our approach can be a supplement to agent communication.

Acknowledgement

The first author's work was done during her visit to Institute for Infocomm Research, Singapore, and funded by the National Science Council of Taiwan under the contract of NSC 93-2917-I-009-001.

References

[1] F.M. Assis Silva, R. Popescu-Zeletin, An approach for providing mobile agent fault tolerance, in: K. Rothermel, F. Hohl (Eds.), MA'98, LNCS, vol. 1477, Springer-Verlag, Berlin Heidelberg, 1998, pp. 14–25.

[2] E. Bierman, E. Cloete, Classification of malicious host threats in mobile agent computing, Proceedings of SAICSIT, 2002, pp. 141–148.

[3] J. Cheng, V. Wei, Defenses against the truncation of computation results of free-roaming agents, Proceedings of ICICS 2002, LNCS, vol. 2513, 2002, pp. 1–12.

[4] J. Claessens, B. Preneel, J. Vandewalle, (How) Can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions, ACM Transactions on Internet Technology 3 (1) (2003 (February)) 28–48.

[5] O. Esparza, M. Soriano, J.L. Muñoz, J. Forné, A protocol for detecting malicious hosts based on limiting the execution time of mobile agents, Proceedings of 8th IEEE International Symposium on Computers and Communication, 2003.

[6] D. Jahansen, K. Marzullo, F.B. Schneider, K. Jacobsen, D. Zagorodnov, NAP: practical fault-tolerance for itinerant computations, Proceedings of the 19th IEEE International Conference on Distributed Computing Systems Austin, Texas, U.S.A., 1999 (June).

[7] N.M. Karnik, A.R. Tripathi, Design issues in mobile-agent programming systems, IEEE Concurrency 6 (3) (1998 (July–Sept.)) 52–61.

[8] G. Karjoth, Secure mobile agent-based merchant brokering in distributed marketplaces, in: D. Kotz, F. Mattern (Eds.), ASA/MA 2000, LNCS, vol. 1882, 2000, pp. 44–56.

[9] N. Karnik, A. Tripathi, Agent server architecture for the Ajanta mobile-agent system, Proceedings International Conference Parallel and Distributed Processing Techniques and Applications, CSREA Press, 1998, pp. 63–73.

[10] P. Kocher, On certificate revocation and validation, Proceedings of 1998 Financial Cryptography, LNCS, vol. 1465, 1998 (February), pp. 172–177.

[11] S. Micali, Certificate revocation system. US Patent 6292893, September 2001.

[12] S. Pleisch, A. Schiper, Fault-tolerant mobile agent execution, IEEE Transactions on Computers 52 (2) (2003 (February)).

[13] K. Rothermel, M. Strasser, A protocol for preserving the exactly-once property of mobile agents, Technical Report 1997/18, University of Stuttgart, Department of Informatics, October 1997.

[14] K. Rothermel, M. Strasser, A fault-tolerant protocol for providing the exactly-once property of mobile agents, Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, 1998 (October), pp. 100–108.

[15] M. Shiraishi, T. Enokido, M. Takizawa, Fault-tolerant mobile agents in distributed objects systems, Proceedings of the ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003 (May), pp. 145–151.

[16] L.M. Silva, V. Batista, J.G. Silva, Fault-tolerant execution of mobile agents, Proceedings of International Conference on Dependable Systems and Networks (DSN 2000), 2000 (June), pp. 135–143.

[17] M. Strasser, K. Rothermel, Reliability concepts for mobile agents, International Journal of Cooperative Information Systems 7 (4) (1998) 355–382.

[18] J.E. W, Mobile agents, Tech. Report, General Magic, Los Angeles, 1995.

[19] J. Zhou, F. Bao, R. Deng, An efficient public-key framework, Proceedings of 2003 ICICS, 2003 (October), pp. 88–99.

[20] J. Zhou, J.A. Onieva, J. Lopez, Protecting free roaming agents against result-truncation attack, Proceedings of 60th IEEE Vehicular Technology Conference Los Angles, California, 2004 (September).

Min-Hua Shao received her PhD in Information Management from National Chiao Tung University, Taiwan, in 2005. She currently works in the Institute for Infocomm Research for a 1-year research collaboration on information security since the summer of 2004. She was a lecturer of business administration in Chung Kuo Institute of Technology, Taiwan. Her current research interests include information security management and financial services such as Internet banking and payment systems in electronic commerce.

Dr. Jianying Zhou is a senior scientist at Institute for Infocomm Research (I2R) and heads the Internet Security Lab. Before joining I2R, he worked in China, Singapore, and USA. He was a security consultant at the headquarters of Oracle Corporation and took an architect role on securing e-business applications. He was a project manager at Kent Ridge Digital Labs and led an R&D team to develop network security technologies. He was a post-doctoral fellow in National University of Singapore and was involved in a strategic research programme on computer security funded by National Science and Technology Board. He was formerly employed in Chinese Academy of Sciences and played a critical role in a couple of national information security projects. Dr. Zhou obtained a PhD degree in Information Security from University of London, an MSc degree in Computer Science from Chinese Academy of Sciences, and a BSc degree in Computer Science from the University of Science and Technology of China. His research interests are in computer and network security, cryptographic protocol, digital signature and non-repudiation, mobile communications security, public-key infrastructure, secure electronic commerce, and virtual private network. Dr. Zhou is actively involved in the academic community, serving international conference committees and publishing papers at prestigious technical conferences and journals. He is a world-leading researcher on non-repudiation and authored the book *Non-Repudiation in Electronic Commerce* which was published by Artech House in 2001. He is a director in the Board of International Communications and Information Security Association. He is a co-founder and steering committee member of International Conference on Applied Cryptography and Network Security and served as program chair of ACNS 2003 and general chair of ACNS 2004. He received National Science and Technology Progress Award from State Commission of Science and Technology in 1995 in recognition of his achievement in the research and development of information security in China.