Migration Using Reordering Recovery in Wired/Wireless Networks[*]

이 동 춘[**]

요 약

유/무선 망에서 통신 노드 간에 통신 실패로 이동 에이전트는 비록 망에서 일시적인 정보 서비스를 이용 할지 모르지만 모든 전송 정보가 블록되고 만다. 이러한 문제를 해결하기 위해 본 논문은 모바일 에이전트가 원할 한 전송을 보장받기 위한 경로 재순서 방법을 제안한다.

Migration Using Reordering Recovery in Wired/Wireless Networks

Dong Chun Lee[**]

ABSTRACT

Due on failures of communication nodes for the wireless and wired networks, mobile agents may be blocked even if there is available service in the networks. To solve it, we propose migration policy with reordering of the paths to guarantee the migration of mobile agents and the paper will provide the extension with the autonomous migration of mobile agents.

Key words : Mobile Agent, Reordering of the Path, Migration, Wired/Wireless Networks

---

* This work was supported by Howon University, 2007.

** Dept. of Computer Science, Howon University, Korea

1. Introduction

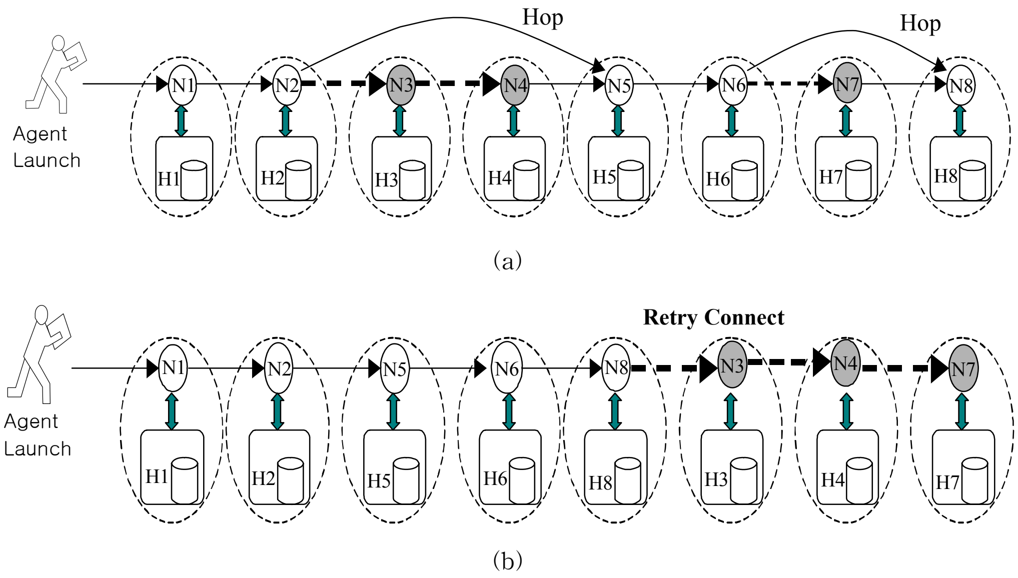
Most of mobile agent systems provide a virtual place for migrating mobile agents under our basic ideal condition that there are no faults on the systems or nodes, or include relevant protocols. While a mobile agent is launched to specific nodes/hosts according to relevant routing schedules [4, 6-8], it is possible to happen some problems about migration of mobile agent if the host happens an accident within where the agent visits and executes. For example, when the node in the middle of the routing path happens a fault on network, the agent may infinitely wait at message queue of the current host in order to connect with the right next node. In other case, even if the agent has migrated successfully in autonomous running process of mobile agent, the agent may not work its own mission when there are no service interfaces among resources that specific host must support, e.g. database. In this situation, it causes low mobile reliability and critical confusing in the e-commerce. This leads to problem of the mobile agent's life span.

Typically, ORB [11, 12] implements distributed garbage collection in order to delete objects having no more references. Voyager [11] provides five policies for mobile agent's life cycle. Mole [2] supports the shadow protocol for orphan detection and successful termination for agents in mobile agent systems. The protocol is for detecting and processing what occur any fault on migrating mobile agents. However, it does not provide to guarantee migration reliability of mobile agent. There is a simple protocol using transaction message queue [10, 13], which is a procedure that the sender puts messages in the queue and receiver gets messages. There is also the same problem as the process of autonomous mobile agent in that it does not include facilities for monitoring the progress of an agent's execution. For example, assume that there is an agent in input queue of a host and the node's error occurs before the agent moves to queue of next node. Then the agent is blocked until that the node is recovered. This situation differs from problem in client/server. Mole [2, 12] provides a fault tolerant protocol to support effective way for 'exactly once' migration using voting and selecting protocol as copying mobile agent to all nodes. Nevertheless, inter-monitoring facility and communication between each observer's node except worker node is needed and assumed that there is no fault in network connection.

2. Reordering of the Whole Path

We will mention an enhanced policy for the agent system to support adaptive migration of mobile agents even if it may happen to a certain failure on nodes or hosts on a whole clustering of the wireless & wired computing networks. The policy is adapted 'fault types' such that agents are not able to migrate more continuously.

The mobile agent may be impossible to migrate to the destination node by the fault of a node or the crash of a host. (Figure 1) (a) supposes that there is a migration path corresponding with an agent's routing schedule and some faulty nodes, such as N3, N4, and N7. An agent migrates and executes from node N1 to N2 se-



(Figure 1) The migration paths before and after meet with faulty nodes based on reordering

quentially, but it is blocked at the host of node N2 until the node N3 is recovered. If the node N3 dose not recovered, the agent may be orphaned or destroyed by the particular host. To solve this situation is for the agent to skip the faulty node N3 that includes on the migration path, and to move the address of node N3 back to the last one of the migration path. Hence, the node N2 successfully connects the next other node N4 without any fault, that is, the agent does not stop processing. In this (Figure 1) (a), node N4 also has a particular fault. Therefore, node N2 hops the right consecutive next one of node N4. As the same method is also apply to other nodes, the agent's migration path has reordered. That is, despite of any particular faulty nodes, the agent tries to connect subsequent nodes for the migration touring. This solution changes the previous arranged migration path by connecting with normal nodes except that some nodes have the particular fault. Afterward, the agent retries to connect each certain fault node after it waits for the timestamp assigned by the mobile agent system. If the certain faulty node is recovering by the timestamp, the agent will succeed in migrating to the destination node. Otherwise, the address of the faulty node will be discarded. Since the agent may be apt to loophole, we will give a restriction against the number of reconnection times. (Figure 1) (b) shows that a whole re-arranged migration path for the mobile agent is changed by this policy.

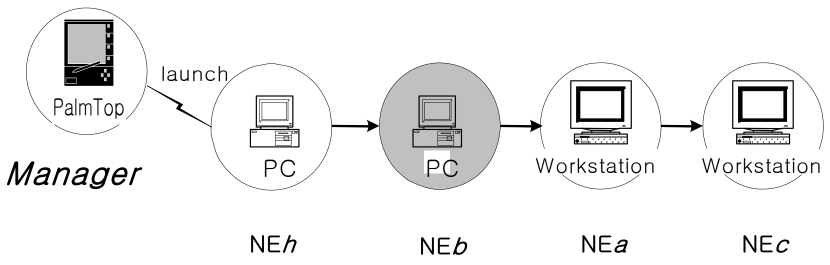
The path reordering executes to connect and communicate with the mobile agent system. If the agent doesn't connect the destination node, it will succeed with connecting the right next node. After the failed address is moved to the last one of routing table, it will be retried to connect about the node. When it does reconnect each failed node, it does wait as much the timestamp assigned by the mobile agent system to re-connect. If it passes over the timestamp, it does ignore this address, and repeat to connect the next fault node. We assign a limitation for reducing network overhead that mobile agent can just try two times to connect the failed node, that is, a mobile agent may occur infinitive looping for just connection. Although it is connected, it is applied equally the same as that way if each host of node errors the mobile agent system. In this way, algorithm 1 offers automatically to reorder the migration path when the mobile agent can not migrated to the next hosts due to the faulty nodes.

〈Algorithm 1〉

```
Algorithm Reordering of the Path

For each agent's routing-table
 {
 extract a target address and fail_checked
 information;
 // multicasting the signal to eliminate that
    clones if the agent has no more than the
    destinations

 if (no more a target address)
     backward multicasts 'Agent_Fire' signal to
     successful_ target nodes ;
 // noticing of the flag to re-entering for the
    failure of some nodes

 if (exist a fail_checked_address) { // check
     whether connect more than two times or not
    wait the agent during some system_
           timestamp ;
    try to connect Socket to the address ;
    if (success) {
           call goAgent ;
                      exit ;
    } else {
           notify to user the address is
                    unavailable ;
       ignore the address ;
    }
 }
 // trying to connect the destination node the
    agent starts to migrate.
    else if (not a fail_checked_address) {
      try to connect Socket to the
           destination node ;
      if (success) {
         call goAgent ;
         exit ;
      }
  // if the agent does not connecting or
     migrating, set the flag and notify to user,
  // re-ordering the path in the routing-table at
     the same time.
     else {
       notify to user ;
       move the current failed_address
           to last in the routing-table ;
       set the fail_checked information ;
     }
   }
}
```

5. Implementation

The proposed policy is implemented a model of mobile agent system developed using Java language. As shows in (Figure 2), the mobile agent system consists of Graphic User Interface, Agents Mobile Service component, Agents Execution Environment component, and Agents Repository to provide the naming transparency of agents. In addition, it may be executing one more systems within a host or a client. Further details refer to [10, 11].



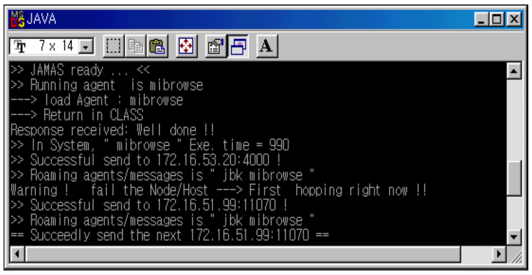
(Figure 2) Routing path having a fault NE$_b$

We show to experiment with an agent which manages some Network Elements (NE). The following figures show the progress that the sample agent as a role of MIB (Management Information Base) browser is migrated and executed according to the routing schedule.

(Figure 2) depicts the routing path of the sample agent such as NE*h* → NE*b* → NE*a* → NE*c*, and we assume faulty at the host NE*b*.

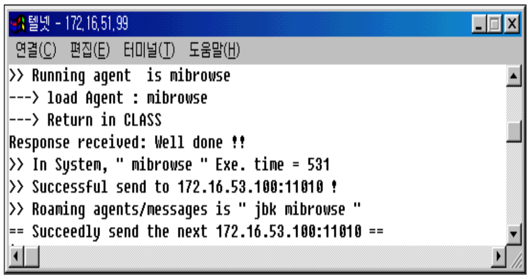
The network manager fetches the prepared agent and specifies routing addresses of it to migrate. So, clicking the 'Go' button on the manager's window (Figure 2) to launch it, the agent starts on a tour to get the MIB information of each NE on behalf of the network manager.

(Figure 3) realizes executions of the agent at each NE. (Figure 3) (a) as a screen capture of the host NE*h*, shows hopping by connection failure at the next NE*b* after the launched agent normally progresses. That is, due to fail the host, the agent passes to next one. Thereafter, (Figure 3) (b) and (Figure 3) (c) capture executing of the agent at the hosts NE*a*, NE*c*. Then it is adapted to the proposed policy. Therefore, the agent has toured for all nodes having no faults before that it does re-connect with the fault nodes.



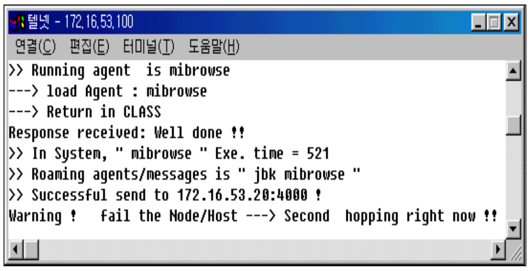
```
>> JAMAS ready ... <<
>> Running agent  is mibrowse
---> load Agent : mibrowse
---> Return in CLASS
Response received: Well done !!
>> In System, " mibrowse " Exe. time = 990
>> Successful send to 172.16.53.20:4000 !
>> Roaming agents/messages is " jbk mibrowse "
Warning !   fail the Node/Host ---> First  hopping right now !!
>> Successful send to 172.16.51.99:11070 !
>> Roaming agents/messages is " jbk mibrowse "
== Succeedly send the next 172.16.51.99:11070 ==
```

(a) Executing at the NE*h*



```
>> Running agent  is mibrowse
---> load Agent : mibrowse
---> Return in CLASS
Response received: Well done !!
>> In System, " mibrowse " Exe. time = 531
>> Successful send to 172.16.53.100:11010 !
>> Roaming agents/messages is " jbk mibrowse "
== Succeedly send the next 172.16.53.100:11010 ==
```

(b) Executing at the NE*a*



```
>> Running agent  is mibrowse
---> load Agent : mibrowse
---> Return in CLASS
Response received: Well done !!
>> In System, " mibrowse " Exe. time = 521
>> Roaming agents/messages is " jbk mibrowse "
>> Successful send to 172.16.53.20:4000 !
Warning !   fail the Node/Host ---> Second  hopping right now !!
```

(c) Executing at the NE*c* and attempting migration of the second at the NE*b*

(Figure 3) Fault-tolerable executions of a mobile agent at each NE

Therefore, the proposed scheme of reliable migration for mobile agents ensure the persistency of computation to preserve autonomous mobility and information of state for agents though there are some faults of nodes or hosts on the routing schedules.

4. Conclusion

In this paper we described a fault-tolerable policy by introducing the path reordering to ensure the migration of mobile agents. The proposed policy not only affords to avoid faults of communication nodes or hosts of mobile agents, but also affects to agents' life span. Future work will investigate for the agent groups with distributed event services.

Reference

[1] K. A. Baharat and L. Cardelli, "Migratory Applications", Proc. of the 8th Annual ACM Symp. on UISTech., 1995.

[2] J. Baumann, "A Protocol for Orphan Detection and Termination in Mobile Agent Systems", TR-1997-09, Stuttgart Univ., 1997.

[3] General Magic, "Odyssey", URL : http://www.genmagic.com/agents/.

[4] IBM, "The Aglets Workbench", URL : http://www.trl.ibm.co.jp/aglets.

[5] B. K. Jeon, "An Efficient Mobile Agent System based on Java Environment", Ph.D. Thesis, Kwangwoon Unv., 2000.

[6] B. K. Jeon and Y. K. Choi, "Design and Implementation of a Mobile System for Java Objects on the Intranet", Jour. of KISS(C), Vol. 5, No. 2, 1999.

[7] B. K. Jeon et al., "Design and Implementation of an Efficient Migration Policy for Mobile Agents", Jour. of KIPS, Vol. 6, No. 7, 1999.

[8] B. K. Jeon and Y. C. Kim, "An Application of Mobile Agents for Efficient Network Management", Proc. of 13th KIPS, Vol. 13, 2000.

[9] B. K. Jeon and Y. C. Kim, "A Reliable Migration Policy using Timestamps for Mobile Agents", Int'l Conf. on Software Engineering Applied to Networking & Parallel/Distributed Computing (SNPD 2000) in France, 2000.

[10] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents", *CACM*, Vol. 42, No. 3, pp. 88-89, 1999.

[11] ObjectSpace Voyager, GeneralMagic Odyssey, IBM Aglets : A Comparison, 1997.

[12] OMG, "Mobile Agent Facility Interoperability Facilities Specification (MAF)", OMG.

[13] A. Puliafito et al., "A Java-based Distributed Network Management Architecture", 3rd Int'l Conf. on Computer Science and Informatics(CS&I 1997), 1997.

[14] S. G. Robert, "AgentTCL : A flexible and secure mobile-agent system", TR98-327, Dartmouth Col., 1997.

Dong Chun Lee

received Ph.D. degree in Computer Science from the Yonsei University at Seoul in 2000. Since 1989 he has been worked in the Department of Computer Science Howon Univ., as a tenure professor. His present research interests concerns the protocol model and the performance evaluation of Mobile Networks, BCN, and Ubiquitous Sensor Networks. Moreover, he researches in the field of Security of Mobile/Ubiquitous Networks. He is an Editor or a Referee of the Journal of Parallel and Distributed Computing (JPDC), Performance Evaluation, European Tran. Telecom. (ETT), Software, Practice and Experience (SP&E), Journal of Interconnection Networks (JION), Journal of High Speed Networks (HSN), Journal of Supercomputing (JS), and Computer Communication (CC), and a Guest Editor of Lecture Notes in Computer Science (LNCS), Asian Journal of Information Technology (AJIT), and Future Generation Computer Systems (FGCS). Also, he is a program committee or organizing committee of seven international conferences, and a member of the IEEE, IEE, ACM, and IEICE. He has published over 60 papers in International Journals and he has Chapter Author 2 International Books (IOS and Springer). He has published profile three Biographical Centers (2000 Outstanding Intellectuals of the 21st Century at International Biographical Centre (IBC) Cambridge in England, Who's Who in the World and Who's Who in Science & Engineering at Marquis Who's Who in USA, and Great Minds of the 21st Century at America Biographical Institute (ABI) in USA). In 2003 he received Award of International Educator of the Year 2003, 21st Century Award for Achievement, Order of Merit, and Order of Distinction at IBC Cambridge in England. In 2004 he received the Best Research Award (Guarantee, $12,000) of the Year 2004, Korea Information Processing Society.