International Journal of Advanced Smart Convergence Vol.4 No.2 38-45 (2015)
http://dx.doi.org/10.7236/IJASC.2015.4.2.38

IJASC 15-2-7

A Super-Peer Coordination Scheme for Decentralized Peer-to-Peer Networking Using Mobile Agents

Won-Ho Chung[*], Namhi Kang[†]

[*,†]*Dept. of Digital Media, Duksung Women's University*

Abstract

Peer-to-Peer(P2P) systems are generally classified into two categories; hybrid and pure P2P. Hybrid systems have a single central index server keeping the details of shared information, so that undesirable effects such as heavy load on that server and lack of fault-tolerance can be caused. Pure P2P causes the other problems such as message flooding and scalability although it shows high degree of fault-tolerance. Recently, mobile agent-based distributed computing has been receiving wide attention for its potential to support disconnected operations, high asynchrony, and thus saving network bandwidth. In this paper, a new scheme of peer coordination is proposed for a decentralized P2P network with self-organizing structure. We deployed mobile agents for incorporating the advantages of usage of mobile agents into our P2P network. Proposed P2P network has both advantages of hybrid and pure P2P. The problems of heavy load on the server and lack of fault-tolerance are improved by using multiple special peers called super-peers. And the problems of pure P2P can be reduced by using mobile agents.

Keywords: *Mobile Agents, Overlay Network, Peer-to-Peer, Super-Peers*

1. INTRODUCTION

Since Napster, the first P2P application system, was born with the purpose of sharing music files distributed over Internet, many other systems with various purposes have been developed. They are in general classified into two categories; hybrid P2P like Napster[1] and pure P2P like Gnutella and Freenet[2][3]. The former has a single central directory server which contains all the indices of shared files of each peer node, so that file search is carried out in that server. Many requests of finding desired files cause heavy load on the directory server and lack of fault-tolerance[4]. In the latter, there is no special directory server. However, it has some other disadvantages of message flooding by many ping and query messages, and of duplicated searching of desired files. Also, some IP addresses of interested peers should have been initially pushed manually in each peer node constituting the P2P network. It is not easy for pure P2P to apply to a practical distributed computing model[5]. Therefore, a new scheme of reducing those problems of hybrid and pure P2P is necessary to improve the performance.

Recently, distributed processing based on mobile agents has been receiving wide attention for its potential to support disconnected operations, high asynchrony, and thus saving network bandwidth. It provides a unified computing paradigm including those of client-server, applet-servelet, and code-on-demand[6][7].

Manuscript Received: Sept. 8, 2015 / Revised: Oct. 2, 2015 / Accepted: Oct. 19, 2015
Corresponding Author: kang@duksung.ac.kr
Tel: +82-2-901-8349
Dept. of Digital Media, Duksung Women's University

Thus mobile agents have been widely applied to various distributed applications such as information searching, filtering and retrieval, electronic commerce and so on[8]-[11].

In this paper, a new decentralized P2P framework, called YAMUS(You-And-Me-US) is developed. It has new schemes of peer coordination and information search using multiple super-peers and mobile agents respectively. It shows both advantages of hybrid and pure P2P. The problems of heavy load and the lack of fault-tolerance due to the central server are improved by deploying multiple super-peers. The problems of pure P2P can also be reduced by using mobile agents. A peer is called a super-peer which has some special functions as well as the functions of a generic-peer. And a peer is called a *generic-peer* which operates as a client only. The decision whether a peer becomes a super-peer or not is automatically made during dynamically organizing P2P network. Any peer can be a super-peer. Since searching information is carried out by mobile agents in a generic-peer side, heavy load problem on a super-peer can be improved. Also, network partition problem can be reduced, and network bandwidth can be efficiently saved by using mobile agents supporting disconnected task operations.

2. OVERALL ARCHITECTURE

Overall architecture of YAMUS is shown in Fig. 1. Each peer node consists of two modules; super-peer module and generic-peer module. It can become a super-peer or a generic-peer. When a peer is coordinated to be a super-peer, both of two modules are activated. Otherwise, only the generic-peer module is activated.

There are 3 types of peers: root super-peer (RSP), voluntary super-peer (VSP), and generic-peer (GP) as shown in Fig. 2. RSP is the special super-peer managing VSP's. It accepts the registration and termination of the peers which are coordinated to be VSP's, and maintains the VSP-list which is a list of VSP's currently registered at the RSP. VSP is also a super-peer managing GP's. It accepts (or rejects) the registration and termination of GP's, and maintains the GP-list which is a list of GP's currently registered at the VSP. It also acts as a GP, and thus each VSP is registered at itself as a GP. In YAMUS, the number of GP's registered at each VSP is restricted, e.g., MAX. The set of GP's registered at a VSP is called a zone which must include a single VSP. Thus, A *YAMUS network* except the RSP can be defined as a set of zones as shown in Fig. 2. YAMUS network can be easily extended by adding zones. However, it should be decided which peer will be a VSP before the construction of a zone. We will use the following notations in our VSP coordination algorithm in section 3. Let *YAMUS network* be Z. Then, Z = {zone(i) | i = 1, …, N}, where zone(i) = {GP(j) | j = 1, …, MAX}.

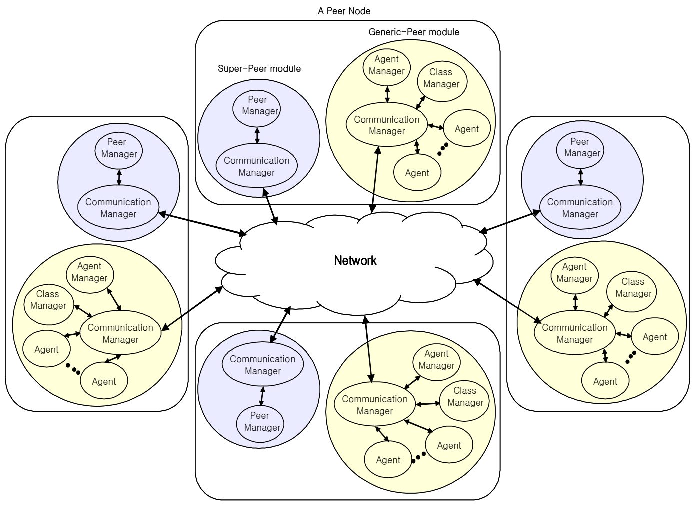


Figure 1. Overall Architecture of YAMUS

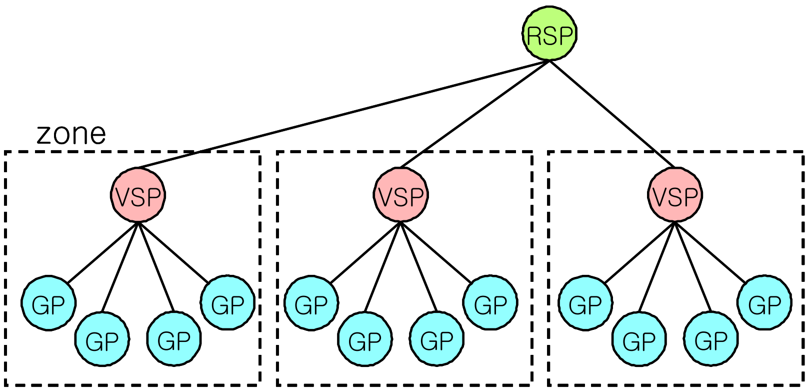


Figure 2. Three types of peers in YAMUS

3. SUPER-PEER COORDINATION AND INFORMATION SEARCH

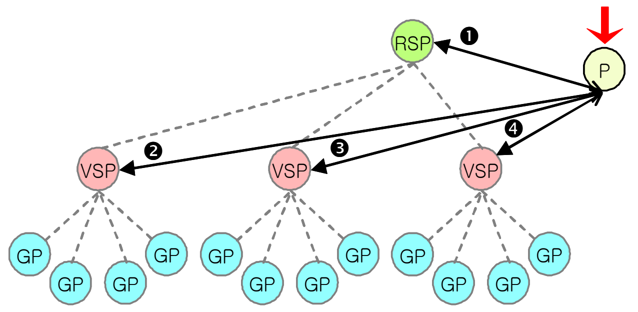
3.1 VSP Coordination

It is common in P2P networks that at least one special peer must exist for organizing the overlay network and initiating information search. That peer is the RSP in YAMUS network. It can be provided as a certain host whose IP is known to all the peers in YAMUS network. That means the RSP plays the role of the initial super-peer from which every search is begun. Every peer except the RSP has to be registered at one of VSP's. Each VSP is coordinated dynamically and automatically during the registration process. It is described in the respect of the system side view, and consists of 3 steps as shown in Fig. 3.

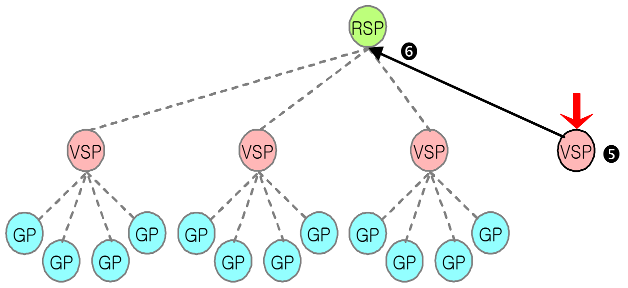
```
STEP-1 : Request and obtain the zone-list, Z from the RSP ;
           /* zone-list is the list of VSP's registered at the RSP */
STEP-2 : for (i=1; i ≤|Z|; i++) { /* execute for each element in the zone-list */
                2.1 : Request a registration to VSP(i) in zone(i) ;
                2.2 : If |zone(i)| < MAX, {
                        register it at VSP(i) as a GP ;
                        exit ; }
         }
STEP-3 : Declare itself as a new VSP and register itself at the RSP, and register itself as a GP on itself ;
           exit ;
```

Figure 3. VSP Coordination Algorithm

A certain peer wishing to be registered, say P, requests the current zone-list, and obtain the list from the RSP. The zone-list is the VSP-list which is a list of VSP's already registered at the RSP. The peer, P requests its registration to one of the VSP's in the zone-list. If it finds a VSP which accepts its request, P is registered at the VSP. The criterion for deciding whether the request for registration will be accepted or not is the maximum number of GP's which can be accommodated in each VSP. It is predefined as a positive integer, say MAX. However, if it can find no VSP which accepts its registration, P voluntarily becomes a new VSP, and requests its registration to the RSP as a new VSP. In this case, the peer, P is registered at the RSP as a VSP. This means another zone is newly constructed. Therefore, the peer in the first definitely becomes the first VSP. And each VSP is registered at itself as a GP. Note that each VSP has at least one GP.



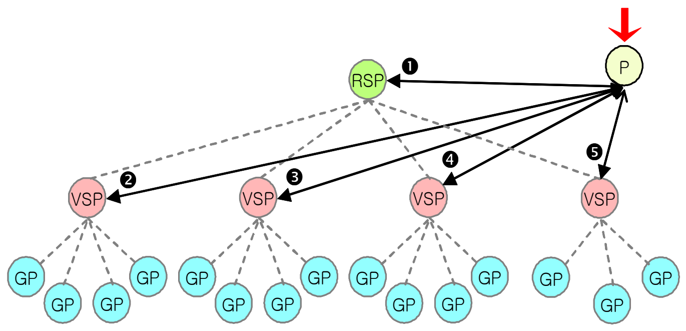
(a) Sequential trial of registration at each VSP, and no VSP accepts it



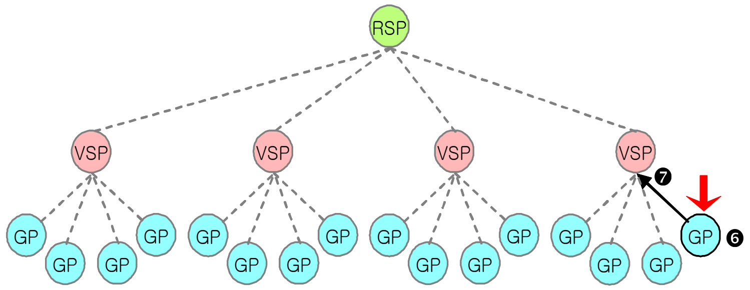
(b) In the sequel , the peer becomes a new VSP

Fig. 4. No VSP accepts the request of registration of a peer, and thus it becomes a new VSP when MAX = 5

Fig. 4 shows an example case that a peer wishing to be activated cannot be registered at any other VSP's, and thus it becomes a new VSP voluntarily. In this example, the maximum number of GP's that a VSP can accept is assumed to be 5, not 4, because the VSP itself is registered at itself as a GP. Fig. 5 is an example case that a peer wishing to be registered is accepted from a certain VSP, and thus it is successfully registered at the VSP as a GP. The zone-list is updated whenever a new VSP is created or terminated.



(a) Sequential trial to each VSP, and a certain VSP accept it



(b) The peer is successfully registered at the VSP

Fig. 5. A VSP accepts the request of a peer, and it is successfully registered at the VSP when MAX = 5

3.2 Searching Information

At the first, a GP must obtain the zone-list from the RSP to search desired information. This is the same as the step in the VSP coordination algorithm. In the next, based on the zone-list, the peer starts searching activities. Note that each peer stores the zone-list obtained during the registration, and obtains it whenever searching activity is start. It is described in Fig. 6.

```
STEP-1 : Obtain the zone-list, Z from the RSP and store it as a predefined file;
           /* zone-list is the list of VSP's registered at the RSP */
STEP-2 : for (i=1; i ≤|Z|; i++) { /* execute for each element in the zone-list */
                 2.1 : Obtain the GP-list kept at the zone(i) ;
                 2.2 : for(j=1; j ≤|GP-list|; j++) { /* this can be parallelized */
                              Send a search agent to GP(j) and
                       /* search agent is a mobile agent */
                   receive the reply from the agent ;
                       }
                 }
           exit ;
```

Figure 6. Searching process

Most of the actions during the search are carried out by sending mobile agents to appropriate GP's. Searching action itself is performed asynchronously at the GP's where mobile agents arrived. Therefore, parallel search can be done by sending more than one search agent. Fig. 7 shows an example of searching when |Z| = 4 and MAX = 5. Although above searching algorithm is described as if it is carried out in sequential manner, searching for GP's belonging to each VSP performed concurrently. In STEP 2.2.2, more than one agent at a time, say k, can be sent to those corresponding GP's. In YAMUS, k = 5 and MAX = 5 for considering our experimental environment. Therefore, in the case of given example, all GP's in a VSP is searched concurrently at a time, and thus search efficiency becomes higher than before.

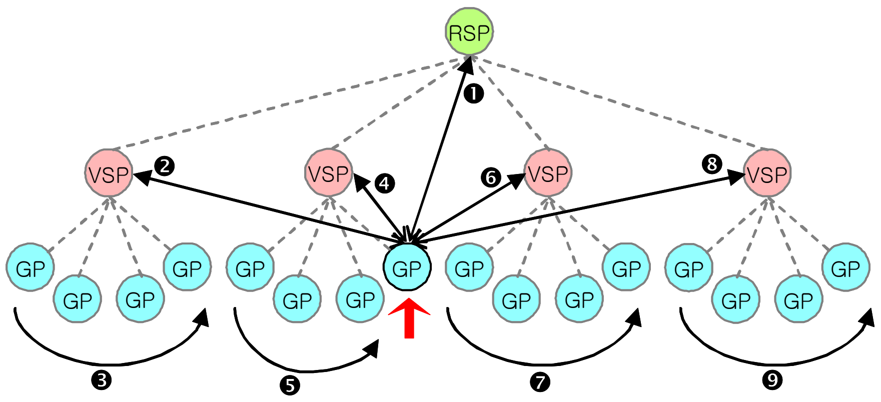


Figure 7. An example of searching for 4-VSP and MAX=5

4. LOAD DISTRIBUTION

A burden of searching desired information is concentrated on a single central server in conventional hybrid P2P systems. In the case of pure P2P it is concentrated on network bandwidth, and thus there exists also a scalability problem. However, YAMUS much reduces and saves the burden of searching and network traffic due to large amount of peers constituting the network.

The RSP does not have any information of files to be searched by network of peers. Hence, any search is not carried out in the RSP. It keeps and manages only the zone-list which is the list of IP addresses of VSP's registered at the RSP. In the same way, VSP does not keep any information of files to be searched. Thus, any search operation is not allowed in the VSP, but it keeps and manages only the list of IP addresses of the GP's registered at the VSP. The maximum number of GP's which can be allowed in a VSP is limited, e.g., MAX. We can adjust the size of a zone by varying the value of MAX.

Let N_V be the number of super-peers including the RSP and VSP's. Then, the message complexity of our VSP coordination algorithm shown in Fig. 3 is $O(N_V)$, because it has no relation with the number of GP's registered at each VSP. Let N_G be the number of GP's in each VSP. Then, the message complexity of a single search is $O(N_V \cdot N_G)$. We regard sending a mobile agent as a message. Therefore, the message complexity for search linearly increases as is proportional to N_V if N_G is given by a positive integer constant. Therefore, the load of search may be fairly distributed over the YAMUS network. There is little burden of searching information on the RSP and VSP, while conventional hybrid and pure P2P networks have much burden on the directory server or network bandwidth respectively. Also, as it is shown in Fig. 6, every search is activated by sending mobile agents, e.g., SearchAgent from each GP to the appropriate peers, and search itself is performed in each GP where the SearchAgent arrived. Mobile agent paradigm allows a disconnected operation, i.e., once a mobile agent arrives at a peer for searching information, it is not essential to keep the connection until a reply of results from the agent is activated. Therefore, allowed network bandwidth can also be saved.

The YAMUS network also shows some fault-tolerant features in the respect of super-peers. As mentioned before, each VSP keeps the updated VSP-list as well as GP-list. Thus, search activities can be kept even if the RSP is shut down during a short range of time. And later the RSP can be recovered with the VSP-list stored in itself as a special file. Also, in spite of that some VSP's do not function normally, those GP's belonging to only the faulty VSP's cannot be searched, but searching activities can be partially maintained for the VSP's in normal operation.

5. AN APPLICATION

It is very uncomfortable to build each peer program for every P2P application. We designed and implemented a useful P2P application which can provide most of representative P2P applications by installing a single peer program. Fortunately, we earned the Bronze prize in the 2nd Java Software Contest which is held by Java Community Organization (JCO) in Korea on Feb. 2004[14]. Currently, five applications are incorporated into the application, and thus we named it "All-in". They are 1) recruit 2) e-market 3) file-share 4) chatting and 5) consult. The All-in application architecture is shown in Fig. 8. It consists of 3 layers except H/W and O.S. layers. They include JVM(J2SE), YAMUS framework and application "All-in", consisting of 5 sub-applications.

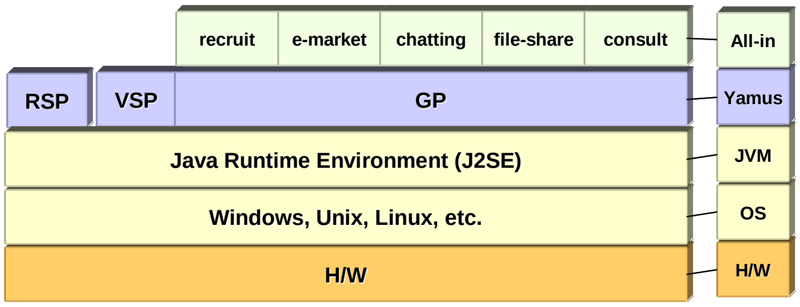
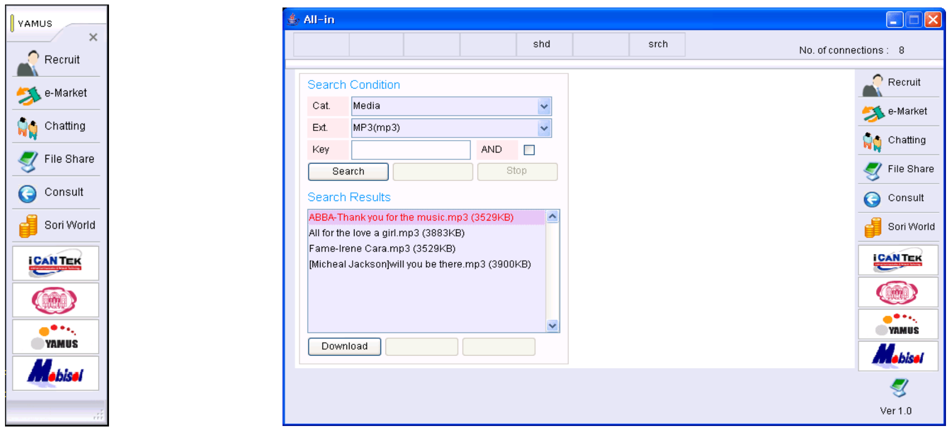


Figure 8. All-In application architecture

All-in is activated by showing a very simple toolbar shown in Fig. 9(a). When you select an application, e.g., File-Share, by clicking the corresponding button on the toolbar, the interface for File-Share is popped up on the screen, where the toolbar is again located in the right-side as shown in Fig. 9(b). If you select other application, the interface corresponding to the selected one will be popped up with the toolbar in the same position. File-Share application has most of functions that typical P2P applications must have. After setting your directory to be shared by pressing the button "shd" in the menu located upper-side (the dialog box for setting the shared directory is omitted), and then by pressing the button "srch" in the same menu bar, some dialog boxes are displayed on the screen as shown in Fig. 9(b).



No. of connections : 8

Search Condition

Cat. Media

Ext. MP3(mp3)

Key AND

Search Results

ABBA-Thank you for the music.mp3 (3529KB)
All for the love a girl.mp3 (3883KB)
Fame-Irene Cara.mp3 (3529KB)
[Micheal Jackson]will you be there.mp3 (3900KB)

(a) Toolbar (b) File-Share interface after pressing "srch" in the menu bar

Figure 9. An example of the YAMUS-based P2P applications, All-In

You must choose some information of files to be searched. When all the required information are provided, simply press the button "Search". Then a mobile agent, SearchAgent will be created and sent to the appropriate peers. And then All-in receives replies from the peers where SearchAgents arrived. Finally, all the files with the selected extension are shown on the result box after the search is finished. When you put a double-click on one of the files displayed on the box (or press the button "Download"), the selected file can also be downloaded into the directory you set.

6. CONCLUSION AND FURTHER STUDIES

This paper proposed the schemes of coordinating super-peers and searching distributed information for a P2P network using mobile agents. We further developed a decentralized P2P framework called YAUMS using the proposed schemes. By doing so, YAMUS reduces the burden of search and shows relatively high fault-tolerance compared to conventional hybrid P2P networks. A mobile agent is a promising way for various distributed applications thank to high asynchrony and disconnected operations saving network bandwidth. Thus, by using the mobile agent for information search, network bandwidth can be saved.

As further studies, we are making effort to extend YAUMS to a fully decentralized P2P network with higher scalability than now.

ACKNOWLEDGEMENTS

This work was supported by the Research Fund, Duksung Women's University in 2014.

REFERENCES

[1] M. Parameswaran, A. Susarla, and A. Whinston, "P2P Networking: An Information-Sharing Alternative," *IEEE Computer*, Vol. 34, No. 7, pp.31-38, 2001.
DOI: http://doi.ieeecomputersociety.org/10.1109/2.933501

[2] Gnutella Development Home Page, *http://gnutella.wego.com/*

[3] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.
DOI: 10.1007/3-540-44702-4_4

[4] B. Yang and H. G.-Molina, "Comparing Hybrid Peer-to-Peer Systems," Proc. Of the 27th VLDB Conference, 2001.

[5] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, 1998.

[6] C. G. Harrison et al, "Mobile Agents: Are they a good idea?," Tech. Report, IBM T.J. Watson Research Center, March 1995.

[7] N. Ivezic, et al, "Agent-based Technologies for Virtual Enterprises and Supply Chain Management," A Draft version for submission *to IEEE Internet Computing*, CTRC, Oak Ridge Nat'l Lab., 1999.

[8] C. Panayiotou et al, "Parallel Computing Using Java Mobile Agents," A working paper, Dept. of Comp. Sci., Univ. of Cyprus, Cyprus.

[9] Won-Ho Chung et al., "A Mobile Agent Scheme with Flexible Reply and Routing for Supply Chain Management," Proc. of Asia-Pacific Conf. on Communications(APCC), 2000.

[10] D. D. Roure et al., "Agents for Distributed Multimedia Information Management," Proc. of 1st Int'l Conf. on the Practical Application of Intelligent Agents and Multi-agents Technology, April 1996.

[11] D. Wong, et al, "Java-based Mobile Agents," *CACM*, Vol. 42, No. 3, March 1999.

[12] Won-Ho Chung and Mi-Yeon Kang, "HUMAN: A Mobile Agent Programming System for Distributed Applications," Proc. of ICIS2002, pp.411-416, Aug. 2002.

[13] JCO, *http://www.javacommunity.org*