The Third International Conference on Internet and Web Applications and Services

# NAT Traversal Method for Multi-Agent-based Meeting Scheduling System

Yusuke Hamada and Shinichi Motomura
Graduate School of Engineering
Tottori University
4–101, Koyama-Minami
Tottori, JAPAN
Email: s032044@ike.tottori-u.ac.jp, motomura@tottori-u.ac.jp

Takao Kawamura and Kazunori Sugahara
Faculty of Engineering
Tottori University
4–101, Koyama-Minami
Tottori, JAPAN
Email: {kawamura, sugahara}@ike.tottori-u.ac.jp

***Abstract*—We previously proposed a meeting scheduling system based on mobile agent technology. The users of our system do not need to input all of their schedules unlike the existing groupwares. When a user intends to call a meeting, he only inputs information about the meeting. On behalf of the inviter, mobile agents move around each invited user's computer to ask whether he can join the meeting and negotiate with him if necessary. In this paper, we propose a NAT traversal method for a meeting scheduling system based on mobile agent technology. In our former system, the users inside a NAT network cannot use the system because mobile agent cannot migrate from outside the NAT network to inside NAT network. For this reason, our proposed system does not work with some network including a NAT router, such as a home network and an office network. To solve this problem, we implemented a new migration method for mobile agents.**

## I. Introduction

The necessity of face-to-face meeting has not decreased at all, even though interactive media such as telephone, facsimile, and email have diffused widely. In general, to organize a meeting is time-consuming routine task.

Some sort of groupware would assist us to organize a meeting. The term groupware refers to software applications, such as Lotus Notes/Domino [1], Microsoft Exchange Server [2], and Cybozu Share360 [3], designed to allow a group of users on a network to work simultaneously on a project. Groupware may provide services for communicating, group document development, scheduling, and tracking.

To utilize a groupware to organize a meeting, all schedule of participants must be managed by the groupware server, therefore all possible participants are requested always to input their schedules into the server. In addition, although existing groupwares have a function to find the date and time on which all participants schedule is open, they don't have a function to negotiate with appropriate participants to open the schedule.

So, we proposed a meeting scheduling system based on mobile agent technology to reduce time and effort on meeting scheduling. Although there are many reports on the agent-based meeting scheduling, especially algorithms or strategies for negotiation among multi-agents [4], [5], [6], [7], [8], there are few on developing an application system which includes the following features:

1) Nobody is requested to input his all schedule into a server before any meeting is intended to call.
2) An agent selects appropriate participants and asks them to open the schedule.

In this paper, we present a NAT [9] traversal method for multi-agent-based system. We aim to construct the proposed system consists of number of user's PCs (hereafter we refer to such a computer as a node) across the Internet, however, communications over a NAT are generally restricted. Therefore, we cannot construct the proposed system consists of nodes, both behind and outside the NAT. For this reason, our proposed system does not work with some networks including a NAT router, such as a home network and an office network. To solve this problem, we implemented a new migration method for mobile agents.

This paper is organized in 5 sections. The design of the proposed system is described in Section 2. The problem and a solution of our system described in Section3. In Section 4, we present discussion about the solution. In Section 5, we present the result of experiments. Finally, in Section 6, we describe some concluding remarks.

## II. Meeting Scheduling System

In this section the design of the proposed meeting scheduling system is presented.

### *A. Main Concept*

The important point in designing a meeting scheduling system is that the means to be used for collecting schedule data and for negotiating. Email and web are popular means to do those things, i.e., requests from the system to users are sent via email, on the other hand, replies from users to the system are sent directly via email or via URL informed by the requesting email. However, we consider that those means are not enough because email is often disregarded. We therefore design the proposed system as a multi-agent system, i.e., we make agents migrate to user's PC for collecting schedule data and for negotiating. A migrated agent opens a window on user's PC for collecting schedule data and for negotiating and it doesn't disappear until the user answers. However, it is necessary to leave email and web as alternative means for the case that a user cannot login over a long period of time when he is on a business trip etc.

Figure 1 illustrates the overview of the proposed system. As shown in Fig. 1, there is a scheduling server which runs always as long as the proposed system provides the meeting scheduling service. A user management agent which stands on the scheduling server manages user information that includes login name, password, IP address of the user's computer, and online/offline status. Login name and password are static while IP address and online/offline status are updated dynamically when a user logins to the system. A user can use any computer on the network because his login name is tied up with the IP address of his computer dynamically. A scheduling agent corresponding to a meeting creates query agents for each participant of the meeting. Each query agent asks one participant's schedule concurrently. The scheduling agent does the rest of the work for meeting scheduling.

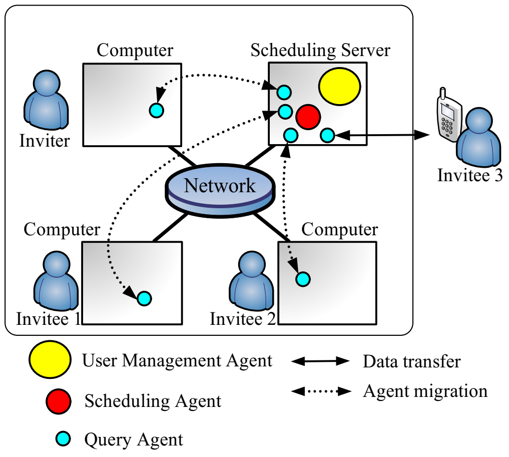


Fig. 1. Overview of the meeting scheduling system.

## B. Schedule a meeting

When a user of the proposed system intends to call a meeting, he/her selects participants from registered users. In addition, he/her provides the further information about the meeting that consists of a timeframe during which the meeting must be held, the duration of the meeting, and the subject of the meeting. Then a scheduling agent corresponding to the meeting is created on the user's computer to arrange the meeting. It migrates to the scheduling server immediately so that the inviter can logout from the system before the meeting is arranged. The IP address of the scheduling server is assumed as known.

Then query agents for each participant of the meeting are created. Each query agent obtains the IP address of a participant's computer from the user management agent, and then migrates to the participant's computer to ask his/her schedule concurrently.

If a user doesn't login to the system when a query agent intends to migrate to his/her computer, the query agent waits a certain period of time on the scheduling server. After timeout, the query agent sends an email to the user informing him/her of a particular URL and waits his/her connection as a web server. The user can tell his/her schedule to the query agent through the URL.

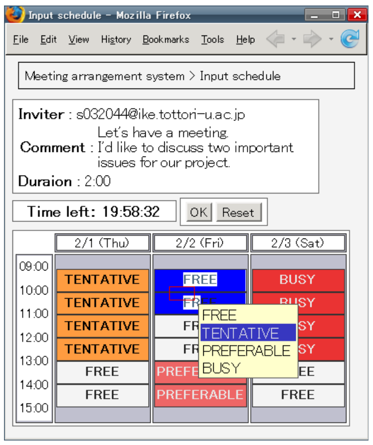
After all the participants' schedules are collected successfully, the scheduling agent arranges the meeting and migrates to the participant's computer to negotiate with the participants if necessary.

At last the scheduling agent notifies the result of the meeting arrangement to both the inviter and the invitees.

## C. Implementation

An implementation of the proposed system has been developed on Java platform by using our mobile agent framework named Maglog [10], [11], [12]. The user interface of the proposed system has been implemented as a Web application using Ajax technology.

Fig. 2 shows a sample screen-shot of query agent migrates to a participant's computer to ask his/her schedule by opening a schedule window. Through this window, one can input his schedule; select a range of hour-cells by holding down the left mouse button and dragging the mouse over the cells; right-click on the cells; choose whether he will be `free`, `tentative`, or `busy` in the pop-up selection box. Only an inviter of a meeting can choose the fourth option `preferable`. Note that a blank cell means that the user is free on that time therefore there is no need to select the `free` option ordinarily. One should select it only when he intends to overwrite other options.


Input schedule - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Meeting arrangement system > Input schedule

**Inviter** : s032044@ike.tottori-u.ac.jp
**Comment** : Let's have a meeting. I'd like to discuss two important issues for our project.
**Duraion** : 2:00

Time left: 19:58:32 OK Reset

| | 2/1 (Thu) | 2/2 (Fri) | 2/3 (Sat) |
|---|---|---|---|
| 09:00 | | | |
| 10:00 | TENTATIVE | FREE | BUSY |
| 11:00 | TENTATIVE | FR | BUSY |
| 12:00 | TENTATIVE | FR | SY |
| 13:00 | TENTATIVE | FR | SY |
| 14:00 | FREE | PREFE | EE |
| 15:00 | FREE | PREFERABLE | FREE |

FREE
TENTATIVE
PREFERABLE
BUSY

Fig. 2. Window for an online user to input schedule data.

## III. NAT Traversal

## A. Problem

In the proposed system, agents migrate among nodes via the scheduling server. When the proposed system is constructed with a scheduling server having global IP address and nodes behind the NAT or nodes outside the NAT, it will work correctly. However, if the proposed system is constructed with a scheduling server having global IP address, nodes behind the NAT and nodes outside the NAT, it will not work. Because scheduling agents and query agents fail to migrate from scheduling server to a node behind the NAT. Therefore, we cannot construct the proposed system consists of nodes, both behind and outside the NAT.

A NAT is widely used in the Internet to share global IP addresses among nodes on private networks. A NAT is suitable for client/server application since a client node behind the NAT can establish outbound connections to a server node which have a globally unique IP address, but the NAT is not suitable P2P application, which needs end-to-end communication, because nodes behind the NAT typically cannot receive inbound connections.

When an agent intends to migrate from one node to other node, the system establishes a TCP/IP connection between the two nodes. However, when the proposed system is constructed with the nodes behind the NAT and the nodes outside the NAT, the nodes outside the NAT cannot establish a connection to the nodes behind the NAT. Therefore, a mobile agent failed to migrate from a node outside the NAT network to a node behind the NAT network. In contrast, the node inside a NAT network establishes a connection to the node outside a NAT network, and receives a response.

### B. Solution

One solution to solve this problem is to set static routing rules to all related NAT routers. If each inside NAT network has only one node, it will be relatively easy. In general, however there are many nodes inside NAT network and those nodes must use identical port number. It is too difficult to maintain those static routing rules by hand. In such cases, UPnP (Universal Plug and Play)[13] is considered helpful. However, it is not realistic to require all NAT routers in the system to have UPnP feature. In addition to that, UPnP will not work with the network such as a NAT router connected to other NAT router.

We construct a new migration method by repeating following steps periodically, as shown in Fig. 3:

1) A node inside a NAT network establishes a connection to a node outside a NAT network, and requests to find mobile agents if mobile agents want to migrate to.
2) If the node outside a NAT network found mobile agents waiting to migrate to the node, they migrate to the node inside a NAT network through a response; otherwise the node inside a NAT sends a response for no mobile agent waiting to migrate to the node exists.

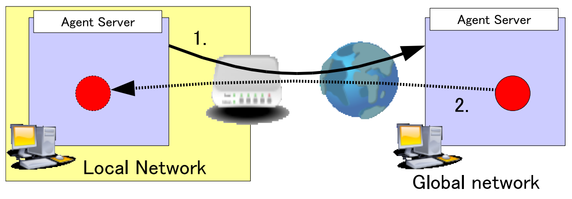


Fig. 3. New migration method for mobile agents.

A new migration method implemented on an agent framework used in our system. And mobile agents in our system reformed using a new migration method if it needs.

## IV. Discussion

Our solution has two restrictions. First, if this migration method runs with short intervals then the system load becomes intolerable, therefore this migration method must run with long intervals. Second, users inside a NAT network can only communicate with well-known nodes outside a NAT network. However, those restrictions are acceptable in our proposed meeting scheduling system. In our meeting scheduling system, all agents have long period to organize a meeting and don't need to migrate to user's node right away. In addition to that, users inside a NAT network only communicate with the scheduling server in our system.

We survey several relational works in this area.

1) Relaying
Some nodes connected with a globally reachable server can communicate with each other by relaying through the server. This is the most reliable technique, however, it consumes the server's processing power and bandwidth.
2) Connection reversal
Connection reversal is a straightforward but limited technique. The basic concept of connection reversal is that a node behind a NAT establishes a reverse connection to a node outside a NAT by an intermediate server when a node outside a NAT communicates to a node behind a NAT.
3) UDP/TCP hole punching
UDP hole punching techniques are proposed for UDP communication over NAT. The basic concept of UDP hole punching is that each node behind a NAT connects a server to make the NAT translation state and registers endpoint information of the node to the server, and then the node connects with each other using the endpoint entry on the server. STUN [14] presents the detailed protocol for UDP hole punching. For TCP communications, some TCP hole punching techniques are also proposed in [15], [16].

Our proposed solution is similar to relaying techniques. To compare with connection reversal, UDP hole punching techniques and TCP hole punching techniques, our proposed solution is simple if target system can accept the restrictions mentioned above.

## V. Experiments

### A. Confirmation of implementation

To investigate the implementation of the new migration method work correctly, we examined our system worked correctly in following environment:

- Two PCs belong to a private network with a subnet range 192.168.2.0/24.
- Two PCs belong to a global network with a subnet range 160.15.35.0/24, including a scheduling server.

- A NAT router for connecting two networks.

The network topology for experiments is shown in Fig. 4.

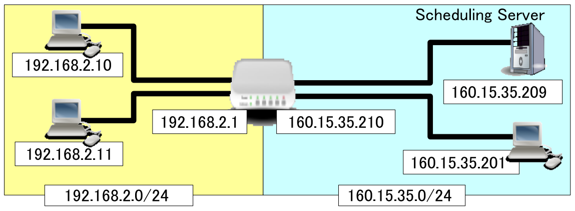


Fig. 4. Network topology for experiments.

To test our system, we use our system in practice with the environment shown as Fig. 4, and we confirmed that our system works correctly in the above environment.

## B. CPU load

We examine the average of CPU load when several nodes inside a NAT network execute almost simultaneously our new migration method to the same node when no agents exist. The experimental environment consists of PCs with Intel Pentium4 3.0GHz processor. They are connected through 1000Base-T Ethernet and are running on Turbolinux 10 operating system whose kernel version is 2.6.0. The version of the Java language runtime environment is 1.5.0, and its heap size is 512MB.

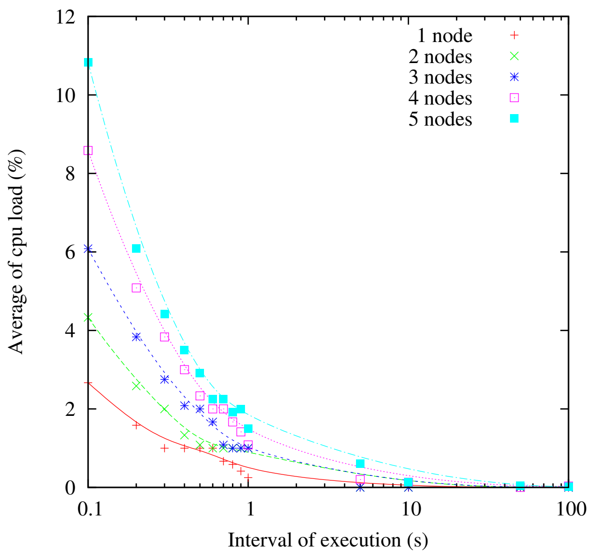


Fig. 5. The change of the average of CPU load when intervals of method's execution increase.

The result is shown in Fig 5. If our new migration method execute with long enough intervals, the average of CPU load becomes almost zero even if the number of nodes increases.

## VI. Conclusion

In this paper, we describe a NAT traversal problem on our former system and implementation of a new migration method for mobile agents to solve the problem. With the improvement of our proposed system, we realize users either inside a NAT network or outside the network can use our system. In other words, users can use our system without consideration about the NAT traversal problem. Therefore we can provide our system for the network including a NAT router, such as a home network and an office network.

To make the proposed system more practical, it is necessary to consider the priorities of participants, i.e., the key person of a meeting must join it; on the contrary, a meeting may be opened without persons with low priority. In addition to that, when the proposed system is used in a large organization, it is necessary to increase the number of schedulable meetings. That will be achieved by a distributed approach in future work.

## References

[1] IBM, "Lotus notes/domino," 2005, http://www-306.ibm.com/software/lotus/.

[2] Microsoft, "Microsoft exchange server," 2005, http://www.microsoft.com/exchange/default.mspx.

[3] Cybozu, "Share360," 2005, http://cybozu.com/.

[4] S. Sen and E. H. Durfee, "On the design of an adaptive meeting scheduler," in *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, March 1994, pp. 40–46.

[5] N. R. Jennings and A. J. Jackson, "Agent-based meeting scheduling: A design and implementation," *IEE Electronics Letters*, vol. 31, no. 5, pp. 350–352, 1995.

[6] H. H. Bui, S. Venkatesh, and D. Kireonska, "Learning other agents' preferences in multiagent negotiation," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, vol. 2, August 1996, pp. 114–119.

[7] L. Garrido and K. Sycara, "Multi-agent meeting scheduling: Preliminary experimental results," in *Proceedings of the Second International Conference on Multi-Agent Systems*. AAAI Press, December 1996, pp. 95–102.

[8] P. J. Modi and M. Veloso, "Multiagent meeting scheduling with rescheduling," in *Proceedings of the Fifth Workshop on Distributed Constraint Reasoning*, 2004.

[9] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," IETF RFC 3022, 1 2001.

[10] S. Motomura, T. Kawamura, and K. Sugahara, "Logic-based mobile agent framework with a concept of "field"," *IPSJ Journal*, vol. 47, no. 4, pp. 1230–1238, 4 2006.

[11] T. Kawamura, S. Motomura, and K. Sugahara, "Implementation of a logic-based multi agent framework on java environment," in *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, H. Hexmoor, Ed., 4 2005, pp. 486–491, waltham, Massachusetts, USA.

[12] S. Motomura, T. Kawamura, and K. Sugahara, "A logic-based mobile agent framework for web applications," in *Proceedings of the 2nd International Conference on Web Information Systems and Technologies*, 4 2006, pp. 121–126, setubal, Portugal.

[13] UPnP Forum, "Welcome to the upnp$^{\text{TM}}$ forum!" http://www.upnp.org/.

[14] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," IETF RFC 3489, 3 2003.

[15] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *Proceedings of the 2005 USENIX Annual Technical Conference*, 4 2005, pp. 179–192, anaheim, California, USA.

[16] S. Guha, Y. Takeda, and P. Francis, "NUTSS: A SIP based approach to UDP and TCP connectivity," in *Proceedings of Special Interest Group on Data Communications (SIGCOMM) Workshops*, Portland, Oregon, USA, 8 2004, pp. 43–48.