MEETING ARRANGEMENT SYSTEM BASED ON MOBILE AGENT TECHNOLOGY

Takao Kawamura, Shinichi Motomura, Kengo Kagemoto and Kazunori Sugahara
Tottori University
4-101, Koyama-Minami, Tottori 680-8552, JAPAN
Email: motomura@tottori-u.ac.jp, {kawamura,kagemoto,sugahara}@ike.tottori-u.ac.jp

Keywords: Meeting arrangement system, groupware, mobile agent.

Abstract: In this paper, a meeting arrangement system based on mobile agent technology is proposed. The users of the 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. Therefore, the inviter's work becomes less compared with using the existing groupwares. We have developed the system with our mobile agent framework on Java platform and confirmed its effectiveness through experiments.

1 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, the following steps are required to organize a meeting.

1. Specify who should join the meeting and a timeframe during which the meeting must be held.
2. Send mails or phone all the participants a number of times to collect their schedules.
3. Merge their schedules and fix the date and time of the meeting.
4. Select appropriate participants, ask them to open the schedule, and return to step 3 if there is no date and time available, otherwise go to step 5.
5. Notify the arranged date and time of the meeting to all the participants.

Those are time-consuming routine tasks, especially steps 3 and 4.

Some sort of groupware would assist us to organize a meeting. The term groupware refers to software applications, such as Lotus Notes/Domino (IBM, 2005), Microsoft Exchange Server (Microsoft, 2005), and Cybozu Share360 (Cybozu, 2005), 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 to input their schedules into the server always. If anyone fails to maintain his schedule on the server, the scheduling function of a groupware becomes useless. 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. That is to say, the above most difficult steps 3 and 4 are left to people.

In this paper, a meeting arrangement system based on mobile agent technology is proposed 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 (Sen and Durfee, 1994; Jennings and Jackson, 1995; Bui et al., 1996; Garrido and Sycara, 1996; Modi and Veloso, 2004), 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.

This paper is organized in 5 sections. The design of the proposed system and an implementation of the system on Java platform are described in Section 2 and 3, respectively. In Section 4, we present the result of experiments. Finally, in Section 5, we describe some concluding remarks.

2 MEETING ARRANGEMENT SYSTEM

In this section the design of the proposed meeting arrangement system will be presented.

When a user of the proposed system intends to call a meeting, he selects participants from registered online users. In addition, he 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 an agent is created to arrange the meeting and collect schedules from both the inviter and invitees. They are requested to classify each hour in the timeframe into three categories: “free”, “tentative”, and “busy”. An agent may ask a user whether he can join a meeting only when the meeting will be held across his tentative hours. In addition to the three categories, an inviter of a meeting can select the category named “preferable” in which he hopes the meeting will be held.

The negotiation procedure of an agent is as follows:

1. Merge all the participants’ schedules and generate a list which contains all the possible continuous hours up to the meeting length. Hours classified as busy by one more participants are excluded.
2. Order the list by the number of hours relating a negotiation, by the number of persons relating a negotiation, and by the inviter’s preference.
3. Exit from this procedure if no negotiation is needed, i.e., the number of hours relating a negotiation of the top element of the list is zero.
4. Retrieve each element of the list and negotiate with the related participants to open their schedules until a negotiation succeeds or the list is empty.

3 IMPLEMENTATION

An implementation of the proposed system has been developed on Java platform by using our mobile agent framework, Maglog(Motomura et al., 2005; Kawamura et al., 2005) .

As shown in Fig. 1, in this implementation, agents described in the previous section are divided two types of agents: scheduling agents and query agents. 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.

In addition to the two types of agents, a user management agent is introduced to manage user information that include 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. Each query agent migrates to the user management computer to obtain the IP address of a participant’s computer at the first, and then migrates to the participant’s computer by using the obtained IP address to ask his schedule. The IP address of the user management computer is assumed as known.

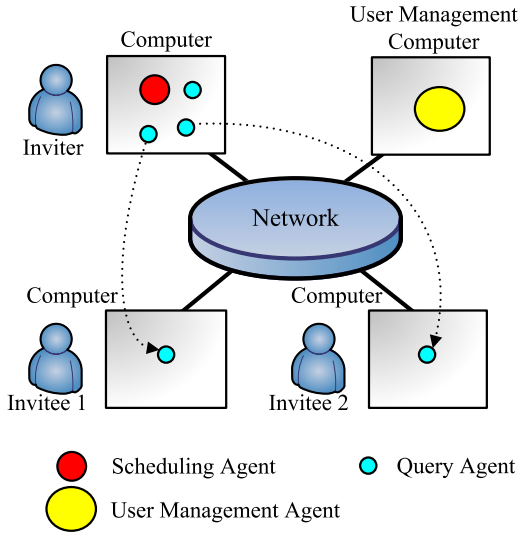

Figure 1: Overview of the meeting arrangement system developed on Java platform.

As mentioned in the above section, when one intends to call a meeting, he needs to select participants and to provide the further information about the meeting that consist of a timeframe during which the meeting must be held, the duration of the meeting, and the subject of the meeting. Figure 2 shows a sample screen-shot of selecting participants and inputting information about a meeting. One can select participants through dragging a node from the tree in the `Available Participants` pane and dropping it into the `Selected Participants` pane. If a dragged and dropped node is leaf then one person corresponds to the node is added to the participants list, otherwise persons correspond to descendant leaves of

the dropped node are added to the participants list at once. An `:off` suffix on a leaf node indicates that the corresponding person is in the offline status, so that he cannot be selected as a participant. When the `OK` button is pressed, a scheduling agent and query agents for each participant are created. Each query agent migrates to one participant's computer and asks his schedule by opening a schedule window as shown in Fig. 3. 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.

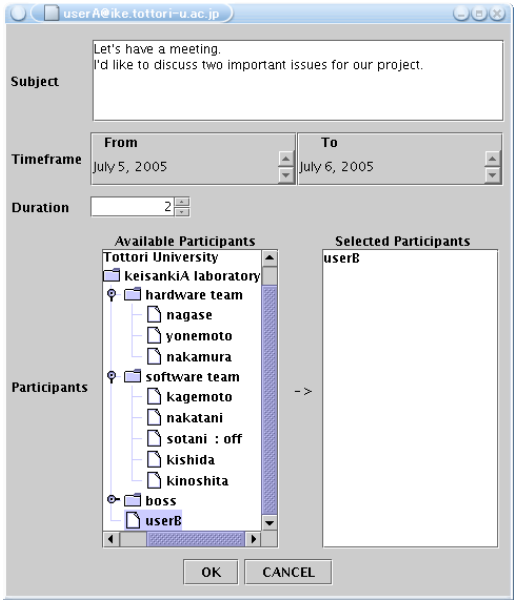

Figure 2: Window for selecting participants and inputting information about a meeting.

4 EXPERIMENTS

This section presents experimental results obtained from the implementation of the proposed system described in the previous section.

The experimental environment consists of two PCs with Intel Pentium4 3.0GHz processor and 1GB of RAM. They are connected through 1000Base-T Ethernet and are running on Turbolinux 10 operating system whose kernel version is 2.6.0.

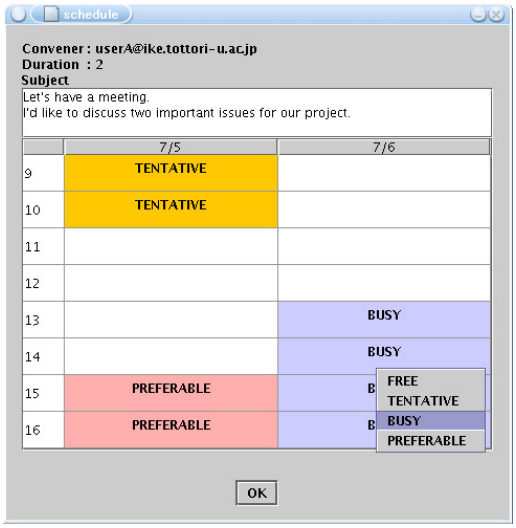

Figure 3: User A inputs his schedule.

As an example, we will examine how a scheduling agent negotiates with participants when user A intends to hold a two hours meeting with user B between July 5th and July 6th. We assume that user A and user B inputted their schedules shown in Table 1. A blank cell in Table 1 means that user A or user B is free on that time.

Table 1: Schedules of user A and user B between July 5 and July 6.

	July 5		July 6	
	user A	user B	user A	user B
9	tentative	tentative		busy
10	tentative	tentative		busy
11				busy
12		tentative		busy
13		tentative	busy	
14		busy	busy	
15	preferable	busy	busy	
16	preferable	busy	busy	

Table 1 shows that some negotiation is needed to hold a two hours meeting between July 5th and July 6th. Table 2 shows durations to be selected as candidates for negotiation in the order of priority. A scheduling agent negotiates with user A and/or user B to open the schedule in that order. Note that it avoids useless negotiation. For example, If one of user A and user B denies to open the schedule on the third duration in Table 2, it does not ask the other user for opening the schedule since the negotiation succeeds only if both two persons agree to open the schedule. In addition, if a negotiation on the third duration fails, it does not intend to negotiate on the fourth duration

in Table 2 since there is no chance that the negotiation succeeds.

Table 2: Possible durations of the meeting on July 5 sorted in the order for negotiation.

Duration	Number of related hours	Number of related persons
11–13	1	1
12–14	2	1
10–12	2	2
9–11	4	2

5 CONCLUSION

In this paper, a meeting arrangement system based on mobile agent technology is presented to reduce time and effort on meeting scheduling. With the proposed system, all an inviter of a meeting has to do is to specify who should join the meeting, a timeframe during which the meeting must be held, the duration of the meeting, and the subject of the meeting. The rest is done by agents, i.e., query agents collect all the participants schedule, and a scheduling agents searches the date and time available for the meeting, negotiates with appropriate participants if necessary, and finally notifies the arranged date and time of the meeting to all the participants. The proposed system simplifies the work not only of an inviter, but also of all participants. Instead of inputting all schedule into a server before any meeting is intended to call, one is requested to input only the schedule between the timeframe of a meeting.

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 prior.

REFERENCES

Bui, H. H., Venkatesh, S., and Kireonska, D. (1996). Learning other agents' preferences in multiagent negotiation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, volume 2, pages 114–119.

Cybozu (2005). Share360. http://cybozu.com/.

Garrido, L. and Sycara, K. (1996). Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 95–102. AAAI Press.

IBM (2005). Lotus notes/domino. http://www-306.ibm.com/software/lotus/.

Jennings, N. R. and Jackson, A. J. (1995). Agent-based meeting scheduling: A design and implementation. *IEE Electronics Letters*, 31(5):350–352.

Kawamura, T., Motomura, S., and Sugahara, K. (2005). Implementation of a logic-based multi agent framework on java environment. In Hexmoor, H., editor, *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 486–491. Waltham, Massachusetts, USA.

Microsoft (2005). Microsoft exchange server. http://www.microsoft.com/exchange/default.mspx.

Modi, P. J. and Veloso, M. (2004). Multiagent meeting scheduling with rescheduling. In *Proceedings of the Fifth Workshop on Distributed Constraint Reasoning*.

Motomura, S., Kawamura, T., and Sugahara, K. (2005). Maglog: A mobile agent framework for distributed models. In *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, pages 414–420. Phoenix, Arizona, USA.

Sen, S. and Durfee, E. H. (1994). On the design of an adaptive meeting scheduler. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, pages 40–46.