

# Middleware Providing Dynamic Group Communication Facility for Cellular Phone Applications

Kouji Nishigaki<sup>†</sup>, Keiichi Yasumoto<sup>†</sup>, Takaaki Umedu<sup>‡</sup>, Teruo Higashino<sup>‡</sup> and Minoru Ito<sup>†</sup>

<sup>†</sup> Graduate School of Information Science, Nara Institute of Science and Technology

<sup>‡</sup> Graduate School of Information Science and Technology, Osaka University

## 1 Introduction

In this paper, we propose a middleware library for efficiently developing distributed cooperative applications consisting of a large number of cellular phone users. Our middleware provides (1) a dynamic group formation mechanism depending on users' locations and preferred subjects and (2) a group communication mechanism called *multi-way synchronization* [1] for multicasting, synchronization and mutual exclusion. Most of Java executors on cellular phones do not support direct communication among user programs. Usable resources are also restricted. Therefore, in our middleware, most of user programs are executed on their servers as agents. Group communication is implemented as inter-process communication on the server, and only the user-interface parts are executed on the cellular phones.

## 2 Proposed Middleware & Implementation

The primitives which our middleware offers are shown in Table 1. *Advertise*, *Participate* and *Disc* are methods for forming a group and for leaving from a group. *Synchronize* is a method for performing communication by multi-way synchronization [1] among the group members.

The construction of groups is executed as follows. (1) The agent executing method *Advertise* broadcasts advertisement messages to all other agents in a server. (2) If an agent that is executing method *Participate* receives the messages, the agent will send back a participation request message. (3) The advertising agent will test the conditions and send back an "ack" message if the conditions are satisfied.

In our middleware, we implement each user program as two separate Java programs: the agent program running on a server and the user interface program (UI) running on a cellular phone.

Each agent on a server includes the main program which the cellular-phone user executes, and is programmed using the methods in Table 1.

method	function
<i>Advertise</i> (ch_list,val_list,cond)	request to advertise group members with a condition
<i>Participate</i> (ch_list,val_list,cond)	request to join the group with a condition
<i>Disc</i> (ID)	release the channel and leave from the group
<i>Synchronize</i> (Event)	exchange data with group members through shared channels by multi-way synchronization

Through UI, a cellular-phone user inputs conditions of group formation and various data, or gets output data transmitted by other users.

We have implemented method **executeApp** to instantiate and invoke the agents from each user's UI. Moreover, we have implemented method **executeMethod** to access the agents from each user's UI. Since the communication with UI and a server was able to use only http, we implemented the protocol imitating RMI on http.

## 3 Experimental Results

In group formation, it is necessary to find a group of agents which satisfy conditions among many users. In our implementation with a common PC, it took less than 1 second to check conditions of 1000 agents.

For the group communication by the multi-way synchronization, 39 data exchanges could be executed per second within a group of 100 members.

We measured with the cellular phone the time for remote method execution by http. The time was about 1.3 seconds.

## References

- [1] ISO : Information Processing System, Open Systems Interconnection, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807, (1989).