2005－UBI－8（16）
2005／6／9

Group Reconfiguration for Community Computing

Jae-Hee Yoon, Jai-Hoon Kim

jhyoon@dmc.ajou.ac.kr, jaikim@ajou.ac.kr
Graduated School of Information and Communication, Ajou University, Korea

ABSTRACT

Community computing and group service become important and popular research issues in ubiquitous computing. Components or agents are combined into group according to their characteristics or specific purposes to cooperate and provide services. Components or agents are scattered over many ubiquitous systems such as sensors, mobile devices and servers and collaborate with group members.

In this paper, we describe reconfiguration of group including group create, group destroy, and join/leave group. We also describe reconfiguration operations between groups. We think that group operations are very useful in many applications in community computing where structure of group devices and services are dynamic.

Keywords: group reconfiguration, community computing, merge group, separate group

1. INTRODUCTION

A principal goal of a ubiquitous computing is providing computational support for the users activities without intrusion [1]. A Ubiquitous computing (Ubicomp) environment composes of heterogeneous devices and services. With the context change groups of devices and services can frequently change. Hence, in Ubicomp environment, applications should incorporate such changes, and provide service to support the activities of users in a non-intrusive way. This requires a design to be more user-centric than device-centric. In real world, users not only interact with each other in community but also interact with devices within the environment. This is so-called a Community Computing environment in which Community Computing should support the process of organizing diverse and ephemeral group of people who are willing to interaction with each other[2, 3]. Compared to groupware studies, Community Computing focuses on a group formation from a variety of people.

In ubiquitous computing environments, the state of resources or services are changed according to context any time. System should also evolve in order to adapt to those changes. Reconfiguration of system according to context should minimize disturbance to normal user action. Dynamic reconfiguration is to allow a system to evolve from one configuration to another configuration during runtime, without restarting the system [4, 5].

The research about group has been studied [6]. This focuses on group for application in distributed computing. Many Applications in Ubicomp environments requires Group Service. As opposed to groups in traditional distributed systems, these groups are dynamically formed and are ephemeral. Further, traditional group management may not be feasible in such environments as Ubicomp environment consists of many types of devices and changes dynamically. Therefore, a technique is necessary for group configuration that is suitable for Ubicomp environment and group reconfiguration that is adapted to such environments.

In this paper, we define that a group includes users, devices and components to support community of users. We describe a group reconfiguration for Community Computing in Ubicomp environments. The group reconfiguration service includes operations for such as group creation, destruction and join/leave group. The Group Reconfiguration Service also includes operation for merging and separation of groups. Since the environment and the state of services are very diverse and dynamically change, the groups for Community Computing in Ubicomp environments can merge or separate. When groups have the same purpose they may merge into one group and while they may separate into two or more groups when the common purpose is lost. Since these operations often occur in Ubicomp environment, diverse application can make use of it.

This paper is organized as follows: Section2 describe related works. In Section 3, we describe our system model and group model for Community Computing. In Section 4, we describe reconfiguration of group and reconfiguration operations between groups. Finally, we offer some concluding remarks in Section 5.

2. RELATED WORK

Many studies have been attempted to build group services in a distributed system environment however few studies have been done for group services in a Ubicomp environment.

The Object Management Group (OMG) has defined the Fault Tolerant CORBA (FT-CORBA) specification [6]. Here, group is used to address a set of replicated objects that supports fault tolerance in FT-CORBA. However in this case, the main point is fault tolerance and usage of group is limited to this purpose. Further, there is no support for a new

This study was supported by a grant of Frontier 21 Project of Republic of Korea.

group service in Ubicomp environment. They have not addressed the issue of flexible configuration and reconfiguration of a group.

Saikoski et al. [7, 8] define group of components in middleware based on OpenORB. They also define that configuration of a group is distributed binding components of various types. They provide that group create, member join/leave and register operations for configuration and reconfiguration of group. However, they do not consider user as a group member and limit group to component level. They do not consider operations of reconfiguration between groups.

Bin Wang et al. [9] presents a model for supporting social groups in a Ubicomp environment. (They explored properties of the social group, and defined what is meant by a social context.) They construct a conceptual model showing the relationship between a social group, and the computing devices used to support those groups. They use device group based on contexts to support user with the perception that the device group is interpreting social contexts. They use CAEG membership management scheme to formation of the context-aware ephemeral device groups. However, there is no support for reconfiguration operation between groups.

Reconfiguration Context Sensitive Middleware (RCSM) [10, 12] is a middleware designed to facilitate application that requires context-aware or spontaneous and ad-hoc communication. It uses ad-hoc ephemeral group, but it dose not focus on reconfiguration of a group.

Dongman Lee et al. [11, 13] propose a middleware infrastructure for Ubicomp environments. They focus on group-awareness, administration-free, and generic middleware for Ubicomp environments. They use a group-context as well as an individual context, and support human-centered environments.

Although these researches propose a solution for groups, they have not addressed the issue of reconfiguration of group. In this aspect, the work presented in this paper improves group services for Community Computing in Ubicomp environments by additional support for group reconfiguration.

3. DESIGN OVERVIEW

This section describes our system model with group support using component technology. We also describe the group model for Community Computing in Ubicomp environments and group reconfiguration design

3.1 System Model

Ubicomp environment can be defined as an environment where people can do their works without considerations that devices exist. Community Computing also supports diverse and ephemeral groups of people in these manner. To support such an environment, our system model is designed around four primary assumptions. First we assume that system should sense environment and convert that information to context. Second, we assume that users collaborate with one or more Ubicomp devices. Third, we assume that the users can communicate over a wireless network. We also assume that the users communicate with each other or devices through Ubicomp devices which are carried by the users and the communication channel is reliable.

3.2 Group Model Concept

In Ubicomp environment, the system for Community Computing must provide group service without user distraction.

A group includes not only devices, components provide services but also users using the services. They can be distinguished by an attribute and a member_type of a Member Profile. The Member Profile consists of tuples < member_id, member_type, session >. Members are merged into a group, combined as multi-object relations, and each member can be simultaneously included into various groups.

A group has a Group Profile that defines group id, purpose and properties of the group. All members in groups can share their Group Profile. When a member wishes to join a group, the members determine whether they can join or not by referencing the Group Profile of the group.

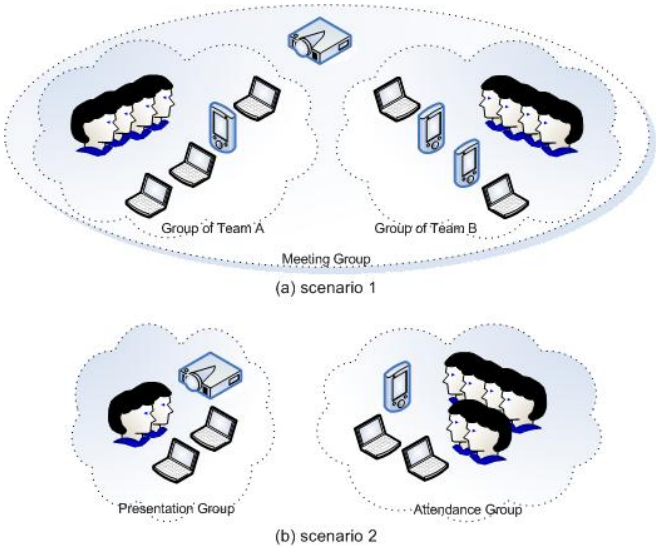

Figure 1: (a) An example of merge groups, (b) An example of separate group

Figure 1 shows two scenarios about merge groups and separate groups. First scenario (Figure 1-(a)) is an example of merge groups and second scenario (Figure 1-(b)) is example of separate group. The detail of scenario is described below.

Scenario 1: In a company, meeting between team A and team B is being held in a meeting room. Before the meeting, two teams have been building different group among each other. While in meeting, both teams have the same purpose. So they are merged into one group (a meeting group) during the meeting. In scenario 1, the two groups do not destroy, and they exist as sub group of the meeting group. There are other case that sub groups will destroy after the groups are merged.

Scenario 2: After the meeting, they hold a technical seminar. At that time, the meeting group is separated into two groups: one is presentation group, other is attendant group.

3.3 Group Reconfiguration Design Overview

Group Reconfiguration Service contains six major components shown in Figure 2:

Reconfiguration Manager: The Reconfiguration Manager is the central component of the Dynamic Reconfiguration Service. The Reconfiguration Manager delegates the Group Factory for creation and destruction of a group. It also delegates the Group Manager for merging and separation of groups. It registers or de-registers groups through interaction with the Group Agent. The Reconfiguration Manager coordinates Group Manager to drive the group to available state to reconfiguration.

Group Factory: The Group Factory is a component in charge of group creation and group destruction. The Group Factory creates a group id, and composes basic components (Group Manager, Group Membership Management, Service Management) which are necessary for Group Service.

Group Agent: The Group Agent provides a registry for a group. It provides group references.

Group Manager: The Group Manager is the central component of the Group Service. A Group Manager is created when a group is created for each of them. It mediates invocation about group reconfiguration. It is also responsible for restricting the behavior of group members during reconfiguration through filtering of requests.

Group Membership Management: The Group Membership Management makes a Member Profile that is a list of member_id and member_type. It provides operations which are related for member to join and leave in a group. It also provides operation for adding and removing members when two or more groups merge or separate respectively.

Service Management: Similarly, the Service Management is a component that manages service provided by a group. It composes services which are provided by each component, a group member, and provides services of group level. It provides operations for registering services and deregister services.

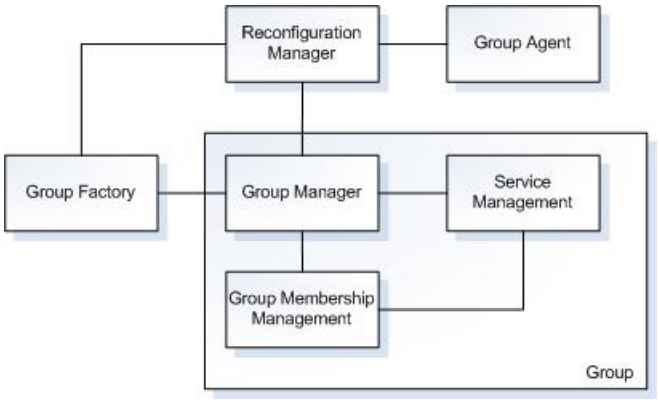

Figure 2: Group reconfiguration design

4. GROUP RECONFIGURATION

4.1 Group Reconfiguration Operations in a Group

1) join member

When a component joins a group to provide service, the Reconfiguration Manager delegates the join of member to the Group Membership Management. The join_member() operation is invoked by the Reconfiguration Manager. The member is added in the group through addtion of id, type, seesion to the Member Profile by the Group Membership Management.

2) leave member

The leave_member() operation is invoked by the Reconfiguration Manager when a member leaves a group. The Group Membership Management deletes the member from the Member Profile, and then notifies the Service Management that a member left the group. The Service Management deregisters the service that is provided by the member who left the group.

3) group creation

The create_group() operation is invoked by the Reconfiguration Manager to create a group. create_group() may be invoked in the course of group creation, group merging or group separation. This operation is delegated by the Reconfiguration Manager for group creation/merging and the Group Manager for the separation to the Group Factory.

The Group Factory creates group_id and composes basic components for the Group Service. The Group Membership Management creates a Member Profile, and manages the member list. The Member Profile contains group_id, member_d and member_ype. The Service Management registers each service that is provided by each member based on information of members joining group. Services that are provided by each member compose services of group level. The services of group level are registered with the Service Manager. The Reconfiguration Manager registers the group on the Group Agent and group creation is completed.

4) group destruction

The Reconfiguration Manager invokes destroy_group() that delegates the Group Factory for destroying the group. The group can completely destroyed when all members finish ongoing invocations. After that, the Reconfiguration Manager deregisters the group from the Group Agent

4.2 Group Reconfiguration Operations between Groups

Configuration unit of group can be not only group member (components, users, devices, etc.) but group itself.

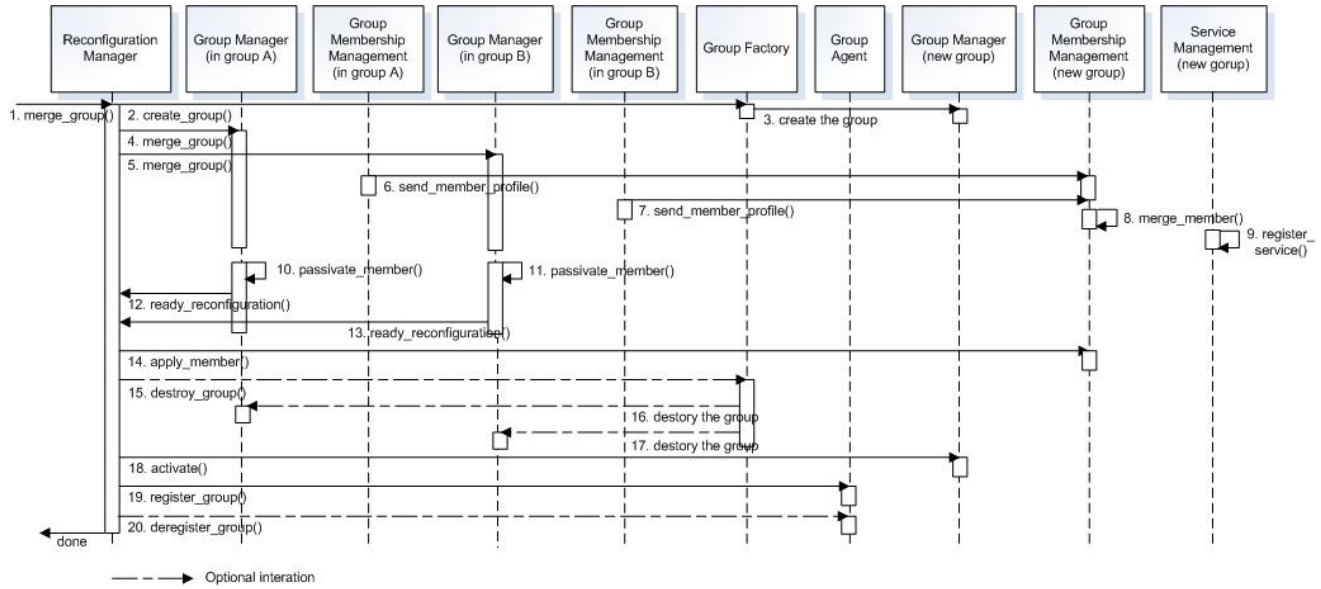

Figure 3: Interaction diagram of merge group

So, two or more groups can merge to provide a common service. On the contrary, one group can separate into two or more groups when they lose common purpose or allocate a part of task to each sub group. Sequence diagram for dynamic reconfiguration is described in reference [4]. But it does not consider group reconfiguration.

1) merge groups

Two or more groups can merge to one group when they have common purpose.

Figure 3 shows the interaction diagram of group merging. The merging of group is started by the Reconfiguration Manager. The steps of group merging are as follows:

1, 2. The Reconfiguration Manager delegates creation of group to the Group Factory.
3. The Group Factory creates a new group. It also creates group id, and defines purpose and properties of the group.
4, 5. When group A and group B merge, the Reconfiguration Manager delegates the merging of groups to each Group Manager.
6, 7. The Group Membership Management of group A sends the Member Profile to the Group Membership Management of the new group. Similarly the Group Membership Management of group B also sends the Member Profile to the Group Membership Management of the new group.
8. Then, the Group Membership Management of the new group merges members through merge the Member Profile which is received from group A and group B.
9. The Service Management of the new group registers services provided by components or devices. It also composes services according to the purpose and properties of the new group, and registers the services.
10, 11. Each Group Managers of group A and group B restricts the behavior of the group members.
12, 13. Each Group Manager of group A and group B notifies to the Reconfiguration Manager when the group members are ready for reconfiguration.
14. The Reconfiguration Manager applies change of members
15. The Reconfiguration Manager delegates destruction of group A and group B to the Group Factory. It is optional interaction because there may be sub group (there may exist some sub group.). In Scenario 1, the groups of team A and team B have merged but they exist as before, and the two groups (group of team A and group of team B) are sub group. In this case, the operation of destroy_group() is not invoked. On the other hand, if the groups (group of team A and group of team B) do not exist after they have merged then the operation of destroy_group() is invoked. Therefore this interaction is optional.
16, 17. If the operation of destroy_group() is invoked then the Group Factory destroy the group(s).
18. The new group (it has merged) is allowed to exhibit active behavior.
19. The group merged is registered with the Group Agent.
20. If the operation of destroy_group() is invoked and the Group Factory destroys the group(s) then the Group Agent is requested to deregister the previous group id of group A and group B. This interaction is also optional.

2) separate groups

One group can separate into two or more groups when the group loses the common purpose or allocates a part of task to each.

Figure 4 shows the interaction diagram of group separation. The separation of group is started by the Reconfiguration Manager. The steps of group separation are as follows:

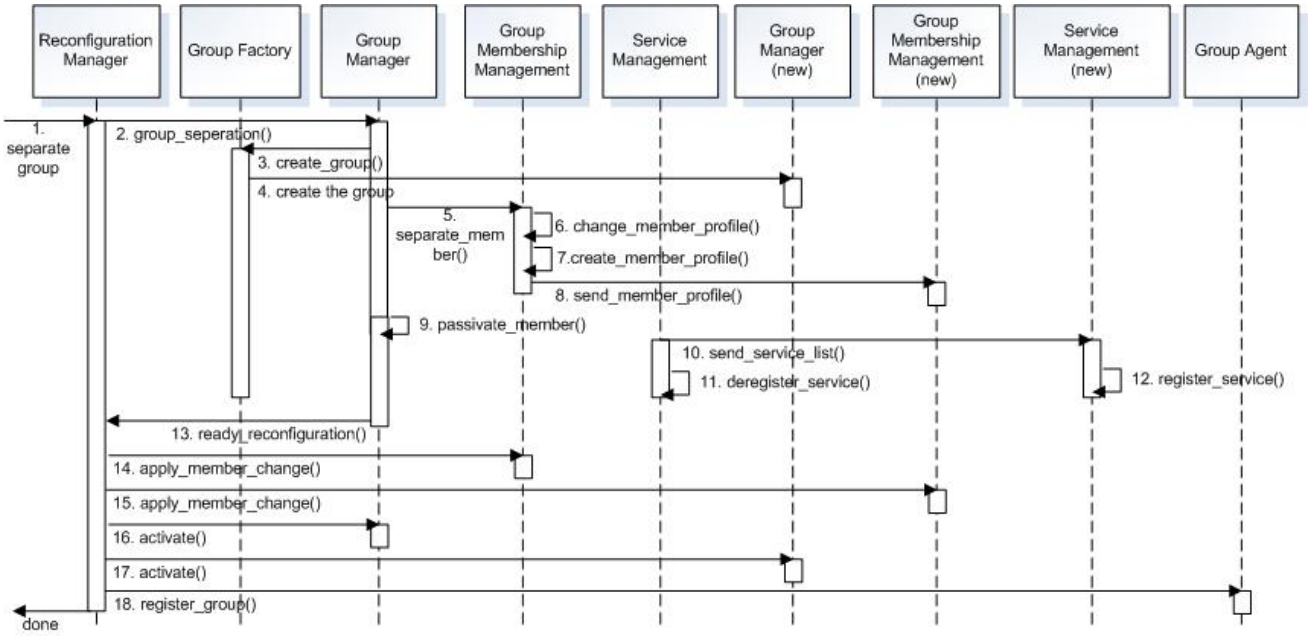

Figure 4: Interaction diagram of separate group

1, 2. Initially, the Reconfiguration Manager delegates the separation of group to the Group Manager.
3. The Group Manager requests Group Factory to create a group.
4. The Group Factory creates a new group and a group id for separation of the group. The Group Factory also composes the basic components (Group Manager, Group Membership Management and Service Management) for Group Service.
5. The Group Manager requests the Group Membership Management to separate members according to groups.
6. The Group Membership Management of original group changes the Member Profile of original group.
7. And the Group Membership Management of original group creates a new Member Profile for the new group which is formed by the separation of group.
8. After that, the Group Manager of the original group sends the new Member Profile to the Group Membership Management of the new group.
9. The Group Manager of the original group restricts the behavior of members for group reconfiguration.
10. The Service Management of the original group makes a list of services according to groups, and sends the Service Management of the new group.
11. The Service Management of the original group deregisters the services included the list of services.
12. When the Service Management of the new group receives the list of services, it registers the services in the list of services.
13. After that, the Group Manager of original group notifies the Reconfiguration Manager when the members are ready for reconfiguration.
14, 15. The Reconfiguration Manager applies change of members to the original group and the new group is formed by separation.
16, 17. The original group and the new group is allowed to exhibit active behavior.
18. Finally, the group formed by separation is registered with the Group Agent.

5. CONCLUSION AND FUTURE WORK

Ubicomp environment enable users to be free from distraction by computers or devices which will be a part of our everyday life. Community Computing also supports diverse and ephemeral groups of people in Ubicomp environments. In this paper we presented a model for supporting group reconfiguration for Community Computing in Ubicomp environment. First, we constructed a conceptual model of group for Community Computing. Then, we a conceptual model was constructed showing the reconfiguration of those groups. Finally, we described reconfiguration operations. The operations include create of group, destroy of group, join a group and leave a group. They also include reconfiguration operation among groups, such as merge groups and separate group. Since the dynamic merging and separation of group are frequently occurred in Ubicomp environments, group reconfiguration is needed for diverse fields of application.

In future, we plan to implement usability container platform to support various levels of reconfiguration: level 0 is reconfiguration of functional factor of component, level 1 is reconfiguration of component and level 2 is reconfiguration of group and service.

REFERENCES

[1] Saha.D, Mukherjee.A. “Pervasive computing: a paradigm for the 21st century”, Computer Vol. 36, No. 3, pp. 25-31 (2003).
[2] Toru Ishida Ed, “Community Computing: Collaboration over Global Information Networks”, John Wiley and Sons (1998).
[3] Toru Ishida, “Towards Communityware”, New Generation Computing, Vol. 16, No. 1, pp. 5-21 (1998) (also appeared in International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-97), pp. 7-21, 1997).
[4] Maarten Wegdam, “Dynamic Reconfiguration and Load distribution in Component Middleware”, (2003).
[5] Mehmet Aksit, Zied Choukair, “Dynamic, Adaptive and Reconfigurable Systems Overview and Prospective Vision”, The 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW’03), pp. 84-89 (2003).
[6] Object Management Group, Fault Tolerant CORBA Specification, v3.0.3, OMG Document formal/04-03-10 (2001).
[7] Katia B.Saikoski, Geoff Coulson and Gordon Blair, “Configurable and Reconfigurable Group Services in a Component Based Middleware Environment”, Proceedings of the International SRDS (Symposium on Reliable Distributed Systems) Workshop on Dependable System Middleware and Group Communication (DSMGC 2000), October 2000, Vol. 15, No.2, pp. 109-126 (2002).
[8] K.B.Saikoski and G.Coulson, “Adaptive Groups in OpenORB”, Proceedings of the 6th Doctoral Consortium on Advanced Information System Engineering (CAiSE'99 DC), Heidelberg, Germany, 14-16 June 1999 (1999).
[9] Bin Wang, J. Bodily, S.K.S.Gupta, “Supporting Persistent Social Groups in Ubiquitous Computing Environments Using Context-Aware Ephemeral Group Service”, Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom’04), March 14-17, pp. 287-296 (2004).
[10] S.S Yau, Karim,F, Yu Wang, Win Wang, S.K.S.Gupta, “Reconfigurable context-sensitive middleware for pervasive computing”, IEEE Pervasive Computing, Vol. 1, No. 3, pp33-40 (2002).
[11] Dongman Lee, “Active Surroundings: A Group-Aware Middleware for Embedded Application Systems”, The 28th Annual International Computer Software and Applications Conference (COMPAC 2004) Vol. 1, pp. 404-405 (2004).
[12] Reconfigurable Context Sensitive Middleware (RCSM), web site: http://dpse.asu.edu/rcsm.
[13] Active Surroundings: A group aware middleware for Ubiquitous Computing Environments, web site: http://cds.icu.ac.kr/druid/research.htm.