Conflict Resolution Method utilizing Context History for Context-Aware Applications*

Choonsung Shin and Woontack Woo
GIST U-VR Lab.
Gwangju 500-712, S.Korea
{cshin, wwoo}@gist.ac.kr

ABSTRACT
In this paper, we propose Conflict Manager to resolve conflicts for context-aware applications in smart home environments. Conflicts arise when multiple users access an application or when various applications share limited resources to provide services. In order to resolve conflicts among users, the Conflict Manager assigns priority to each user so that the user with the highest priority can be selected by exploiting conflict history of users. In addition, Conflict Manager detects and resolves conflicts among applications by utilizing preferences of users and properties of the services. To show the usefulness of the proposed conflict resolution method, we apply the proposed conflict resolution method to ubiHome, a smart home test-bed. The experimental results proved that Conflict Manager enable context-aware applications to offer personalized services to multiple users by resolving service conflicts among applications as well as among users.

Keywords
Context-Awareness, service conflict, context history

INTRODUCTION
The aim of ubiquitous computing is to provide users with intelligent services based on the information obtained from distributed but invisible computing resources. These services do not require any cumbersome interface or leaning procedures for users to use them. Especially context-aware applications offer appropriate services to users by utilizing contextual information of environment including users [1]. This information is obtained from various sensors or computing resources distributed in our daily life. However, conflicts occur in context-aware applications when multiple users share the applications or these applications share the limited resources in environment. Service conflict among users is the scenario when multi-users access an application, and then the application have to choose one user to provide a customized service. As a result, the applications could not make a suitable decision to start a service, and each user may not receive personalized services. Resource conflicts also occur among services if each service attempts to share resources at the same time. Consequently, applications start serving to the users without possessing all the necessary resources and thus may result in unsatisfactory services.

Over the last decade, most research, aimed on resolving conflicts, has been done on smart home and intelligent office. Reactive Behavioral System (ReBa) supports conflict resolution among devices in office environment such as, between electric lamps, display devices, and telephones [2]. RCSM (Reconfigurable Context-Sensitive Middleware for Pervasive Computing), an object-based framework, makes sensors and application services independent, forms ad-hoc communication between them, and delivers the necessary context to the applications [3].

However, context management in the previous research has various limitations when they are applied to multi-user environment with various applications. In the case of ReBa, it is difficult to provide to each user with particular services because ReBa focuses on the service for grouped users by inferring main activities from the environment [2]. In RCSM, context management does not consider shared devices or services because contextual information services are provided only through individual device possessed by each user [3].

In this paper, we propose Conflict Manager to resolve service conflict caused by the use of applications among multiple users and limited resources among multiple applications. The proposed Conflict Manager consists of three parts: i) User Conflict Manager which resolves conflict among users , ii) Service Conflict Manager which resolves conflict among services, and iii) Conflict History Manager which assigns priority to conflicting context by utilizing conflict history. Conflict Manager resolves the conflicts among users by choosing a user having the highest priority. In addition, the proposed Conflict Manager detects and resolves conflicts among applications by utilizing properties of services and relationship among them.

* This works was supported by DSC of Samsung Electronics Co., Ltd., in Korea

This paper is composed as follows. First of all, we introduce service conflicts caused by multiple users and multiple applications in context-aware computing environments. We also classify the service conflicts into three types according to conflict sources. We then describe Unified Context-aware Application Model for ubiquitous computing environment (ubi-UCAM). We then introduce Conflict Manager which resolves services conflicts among users and among applications. Finally, we explain the experimental results of applying this method to ubiHome test-bed.

CONFLICTS IN CONTEXT-AWARE APPLLICATIONS

In context-aware computing environments, various applications provide users with customized services based on users' contexts within a service area. In order to provide the services, the applications require one or more resources, such as display device, sound device, light device, or, etc, according to their properties. Furthermore, in such service environments, the number of users accessing the same applications is not limited.

Unlike single user and single service environment, applications in the computing environment have to respond while considering other applications and various users within a services area. We define such situation as a service conflict. We classify the conflict into three types according to sources of conflicts: service conflicts among multiple users, service conflicts among multiple applications and service conflicts among multiple users and multiple applications. Service conflicts among users are caused due to use of an application by multiple users. In this situation, the application has to choose one customized service. For example, a service conflict arises when users A and B are trying to watch their preferred broadcasts from television service. Service conflicts among multiple applications are caused by providing of services among multiple applications. Due to the conflict, the application cannot provide users with customized responses. For instance, this kind of conflicts occurs when television application and music application start to provide their customized services simultaneously. Service conflicts among users and applications are caused due to the use of multiple services by multiple users. This kind of conflict is similar to the conflict among applications, but the users assigned to the applications are different. For example, a service conflict arises when user A is trying to use a television application while user B is trying to use a music application.

To deal with these conflicts, resolution methods have to resolve the conflict according to sources of conflicts. Furthermore, in order to reflect the change of users' preferences and their environment, the conflict resolution methods must adapt to users and their environment. In this paper, we deal with two kinds of conflicts, i.e. among users and among services, by utilizing conflict history of users as well as user contexts and service profiles.

UNIFIED CONTEXT-AWARE APPLICATION MODEL

In order to deal with service conflicts, we adopt Unified Context aware-Application Model for ubiquitous computing environment (ubi-UCAM) [4]. The ubi-UCAM is a context-based application model to provide users with personalized services by exploiting context in ubiquitous computing environments. In addition, to ensure independence between sensors and services, the ubi-UCAM utilizes unified context represented as 5W1H (Who, What, Where, When, How and Why) [4]. The ubi-UCAM employs different types of unified context based on the role of each context. These include preliminary context, integrated context, conditional context, and final context. Figure 1 shows the overall architecture of the ubi-UCAM.

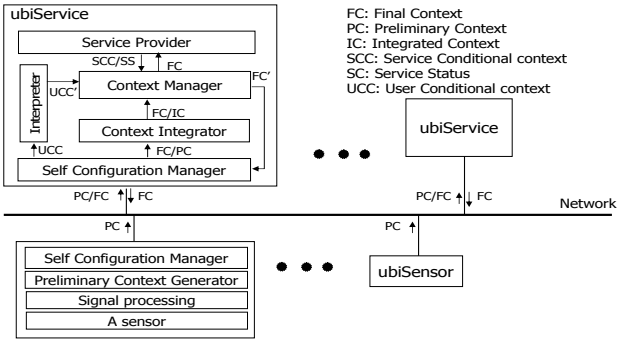

Figure 1. ubi-UCAM

As shown in Figure 1, the ubi-UCAM is composed of ubiSensors, a sensor, and ubiServices, an application to provide a service. Each ubiSensor generates a preliminary context from the features extracted from a physical sensor. It then delivers the preliminary context to ubiServices within a service area. Each ubiService collects preliminary contexts as well as final contexts delivered from other ubiServices within a service area. The ubiService then builds integrated context of each user by classifying the preliminary contexts and final contexts. It searches conditional context from a Hash table, which manages specific service action and condition, corresponding to each integrated context. It generates a final context to be used by Service Provider after resolving conflicts among users and services. Finally, ubiService executes appropriate action with parameters described in the final context. It utilizes application-specified methods which are programmed by application developers.

CONFLIICT MANAGEMENT

In ubi-UCAM, service conflicts occur not only due to multiple users who access ubiServices at the same time, but also due to multiple ubiServices trying to share resources in their surrounding. To resolve service conflicts among users, the proposed Conflict Manager assigns priority to users and chooses the user given the highest priority. In addition, to deal with service conflict among ubiServices, the Conflict Manager detects and resolves conflicts, based on the properties of ubiServices and relationship between

them. Moreover, priority of users and ubiServices are not fixed, but adapts to user's preference and behaviors. Therefore, the Conflict Manager not only resolves conflicts among users and among ubiServices, but also dynamically assigns priority to users and ubiServices.

Conflict Resolution among Users

User Conflict Manager resolves conflicts caused by users who try to use ubiServices within a service area. To resolve the conflict, User Conflict Manager manipulates user contexts in two steps: building a conflict list and selecting a proper user from it. Figure 2 depicts the overall procedure of User Conflict Manager.

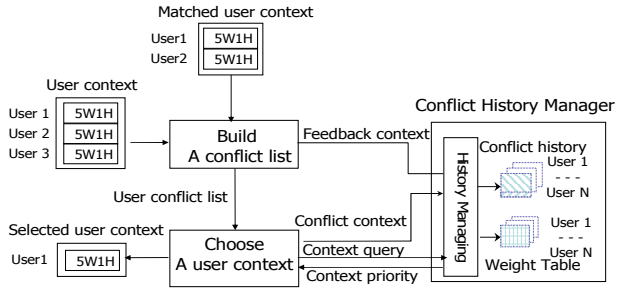

Figure 2. User Conflict Manager

As shown in Figure 2, User Conflict Manager makes a conflict list of matched user context on users who are expected to cause conflict among users, including those who are currently using the service. In this process, users who leave the service area are excluded from the list because we assume they do not want to use the service any more. In addition, user's feedback is also delivered to Conflict History Manager. The context is considered as user feedback if there is user implicit context such as remote controller. In the next stage, User Conflict Manager chooses one user from the conflict list based on user's priority calculated from Conflict History Manager according to user context. In this process, conflicts are handled in several ways according to the number of users within the service area. In the case of one user situation, we know that there is no conflict among users. Therefore, User Conflict Manager just selects the user context as a result of conflict resolution. However, we have to consider the situation when there are more than two users in a service area. In this situation, User Conflict Manager selects the user having highest priority because conflicts may occur. In addition, it notifies the result of conflict resolution to enable Conflict History Manager to store conflict context.

Conflict Resolution among ubiServices

Service Conflict Manager resolves services conflicts caused by multiple ubiServices trying to share resource in the service area. The conflicts are caused by not only a ubiService itself but also other ubiServices. Therefore, Service Conflict Manager deals with the conflict in two ways: conflict caused by other ubiServices and conflict caused by a ubiService itself. Figure 3 shows Service Conflict Manager.

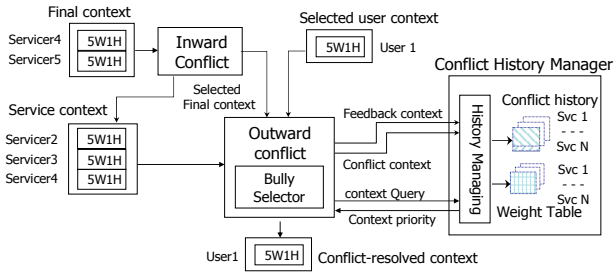

Figure 3. Service Conflict Manager

As shown in Figure 3, Service Conflict Manager creates a context which contains information about the ubiService and its stop action, if resources involved in other ubiServices are the same as those of the ubiService itself. As a result, the application responds to changes of other ubiServices which cause conflict, using final contexts coming from other services. In addition, Service Conflict Manager updates the final context to the final context table. Service Conflict Manager prevents this ubiService causing conflict with other ubiServices. To detect possible conflicts, it checks to see if there are any services using the same resource before delivering the context. Service Conflict Manager compares priority of the service contexts calculated from Conflict History Manager if there are conflict services within a service area. Finally, it sends the conflict-resolved context to Final Context Generator when there aren't any services related to the same resource. In addition, Service Conflict Manager just sends the resolved context to Conflict History Manager to notify the result of conflict resolution.

Service Conflict Manager also deals with the situation when multiple services want to use resources at the same time. This is because ubiServices can respond to the same condition. In the case of this conflict, several ubiServices want to use the same resource. For example, television and movie services can be triggered at the same time when a user enters home. To deal with this situation, we adopt bully algorithm that elects a leader among processes in distributed computing environment. The algorithm chooses a coordinator having the highest priority among processes [7]. In service conflict, the algorithm is used to choose the highest ubiService among ubiServices which try to use shared resources.

Conflict History Management

Conflict History Manager takes charge of maintaining conflict history and determining priority of conflicting context. To efficiently use the limited storage, it only maintains conflict history for a short period of time. In addition, to reflect user preference, Conflict History Manager calculates the priority of conflicting contexts based on Bayes theory which is widely used for

classification or prediction. Figure 4 shows the overall architecture of Conflict History Manager.

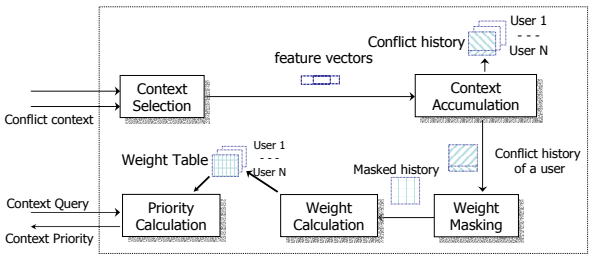

Figure 4. Conflict History Manager

As shown in Figure 4, Conflict History Manager receives feedback and conflicting contexts of users from Conflict Manager. Based on the contexts, Conflict History Manager generates a feature vector containing information about the conflict situation. Afterwards, the feature vector is stored in a history file so that it can be retrieved whenever required. Conflict History Manager then loads the feature vectors, related to a specific user, from conflict history. Conflict History Manager recalculates weights of conflicting contexts based on the feature vectors. In order to obtain the weight, Conflict History Manager applies Bayesian theory to the feature vectors. Equation (1) shows Bayesian theory. In the equation, feature vector X is composed of $(x_1, x_2, x_3, x_4, x_5, x_6)$. Each element of X is mapped to the value of Service type, Location, Time, Gesture, Stress, and Conflicting user. The result of conflict resolution H_j, which is represented by (H_1, H_2) indicates the Target class. Consequently, we obtain probability $P(H_1|X)$, for allowing the current user of a service to continue using the service when conflict arises, by multiplying posteriori probability $(X|H_1)$ and prior probability $P(H_1)$.

$$P(H_j \mid X) = \frac{P(X \mid H_j)P(H_j)}{P(X)} \qquad (1)$$

According to the equation, we assume that a current user of a ubiService will continue using the service in case of a conflict when posteriori priority $P(H_1|X)$ is greater then $P(H_2|X)$. Otherwise, another user uses the service. So, a priority of context is the difference between maximized posteriori probability of $P(X|H_1)(H_1)$ and $P(X|H_2)(H_2)$. Therefore, a weight of each feature is expressed by priori probability of the feature $P(x_k|H_j)=s_{kj}/s_j$. s_{kj} is the number of conflicting contexts having a specific value of s_k within the class H_j class. s_j is the sum of values of conflicting contexts belonging to H_j. Conflict History Manager calculates weights of conflicting contexts of users based on the weight table. The calculated results are updated in hash-table and a weight file for future search.

Conflict History Manager provides priority of the conflicting context based on the weight table when Conflict Manager requests priority for a conflicting context Conflict History Manager retrieves weights of the user, identified by 'Who' context of conflicting context, from the hash-table. Afterwards, it applies the weights to the conflicting context to Equation (2) to calculate posteriori probability. The Conflict History Manager calculates posteriori probability $P(X_i|H_1)$ when a current user will continue using the service, and posteriori probability as $P(X_i|H_2)$ when another user will use it.

$$P(X_i \mid H_j) = \prod_{k=1}^{n} P(x_k \mid H_j) \qquad (2)$$

Finally, Conflict History Manager calculates a priority of the conflicting context. Equation (3) shows the priority of conflicting context. In the equation, $P(X|H_1)P(H_1)$ is the maximized probability of the current user to continue using the service. $P(X|H_2)P(H_2)$ is the maximized probability of another user to use the service. Conflict History Manager delivers the difference of these two probabilities to Conflict Manager as a priority of the conflicting context.

$$Priority\ (X_i) = P(X_i \mid H_1)P(H_1) - P(X_i|H_2)P(H_2) \qquad (3)$$

Based on the conflict history and Bayesian theory, Conflict History Manager adjusts the weight of conflicting context using conflict history of users after conflicts are resolved. It also assigns a priority to conflicting contexts of users based on the weight table when conflicts arise.

IMPLEMENTATION AND EXPERIMENT

We have evaluated the effectiveness of the conflict resolution method based on the ubiHome test-bed. The proposed Conflict Manager selects one among several users when multiple users attempt to access their registered service. In addition, it decides to provide the service when priority of the service is higher than the other services located within a service area. Finally, we also measured accuracy of the proposed method with four family members

Experimental Setup

The proposed Conflict Manager was implemented with J2SDK 1.4™ so that it can be applied to various applications. As shown in Figure 5, we tested Conflict Manager in ubiHome, a smart home test-bed at GIST [5].

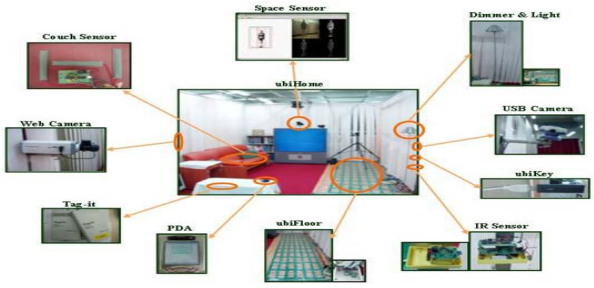

Figure 5. ubiHome test-bed

As shown in Figure 5, we utilized various ubiServices such as, television service, Internet service, music service, movie service, light services, etc. These ubiServices offer customized services to users. In addition to the services, we also exploited various sensors: ubiCouch sensors,

ubiTrack, and ubiRemocon. The ubiCouch sensors, comprised of on/off switches, detect user's behaviors. The ubiTrack is infrared-based location tracking system that tracks users' location [6]. The ubiRemocons are a kind of remote controllers, implemented with Personal Java, to control these services.

Experimental Analysis

In order to measure accuracy of resolution method of the proposed Conflict Manager, we experimented on user conflict in two ways: i) a resolution method based on the Bayesian theory and, ii) a resolution methods having fixed priority. To test two methods, we employed television service that users use in a home environment. While using the television service, family members cause conflicts due to their preferences and its broadcasts. In our experiment, the television service selects a preferred program a user. It decided a specific program of the user who has the highest priority according to each selection strategy when conflicts occurred. The service gathered feedback from users in pre-defined amount of time and judged the accuracy on the selection. The television service counts the number of "incorrectness" and "correctness" of the selection. As the result of the selection, we have built confusion matrix to know how well it works. Table 1 shows the experimental results of the proposed conflict resolution method

Table 1. Confusion matrix for conflict resolution (unit: %)

Users	Father	Mother	Son	Daughter
Father	81	8	4	7
Mother	8	79	7	6
Son	4	3	78	15
Daughter	5	6	14	75

As shown in table 1, the proposed resolution method provides the television service to other users who have lower priority in the conflict resolution having fixed priority. This is because the conflict resolution method assigned priorities to users based on their context. In addition, accuracy of the resolution method was relatively higher than the resolution method having fixed priority. The improvement of accuracy was due to the fact the resolution method reflected the changes of their preference and resolution policy. Therefore, conflict solution could resolve service conflicts caused due to use of services by multiple users.

In addition, we configured properties of services to deal with conflict among services. In our experimental setup, all the services were in the same area. Especially, television, and movie services were operated on the same computer. Based on the properties, we monitored the services in ubiHome in order to observe resource conflicts among services. Table 2 shows the number of conflict among services.

Table 2. The number of conflict among services (unit: %)

Services	Television	Movie	Music	Light
Television	-	33	56	11
Movie	54	-	25	21
Music	72	28	-	-
Light	77	23	-	-

In case of television service, most of the conflicts are related to movie service. The rest of the conflicts are associated with movie and music service. Movie service, which shares sound, light, and display resource, is related to all the services. In particular, conflicts of movie service are mostly due to television service which is accessed by users frequently. Besides, movie service also conflicts with light service since the services use light resource. Music service was related to television and movie service using sound and display resources.

CONCLUSION

In this paper, we proposed the Conflict Manager to resolve services conflicts among users and among applications. In order to resolve conflicts among users, the proposed Conflict Manager maintained the conflict history of users, calculated the priority of user context with Bayes theory, and then selected one user. In addition, Conflict Manager detected conflicts among applications based on properties of each service. These conflicts were resolved with the priority so that the applications provided services without causing conflicts. In our future works, however, we will employ additional applications deal with the conflicts. We will also observe the conflicts with users' behaviors over longer periods.

REFERENCES

1. Anind K. Dey, "Understanding and Using Context. Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing, 5(1),. (2001)
2. Nicholas Hanssens, Ajay Kulkarni, Rattapoom Tuchinda, and Tyler Horton, "Building Agent-Based Intelligent Workspaces," In *ABA Conference Proceedings*, June. (2002)
3. S. S. Yau, F. Karim, Y. Wang, B. Wang, and S.Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *IEEE Pervasive Computing, joint special issue with IEEE Personal Communications*, 1(3), pp.33-40, July-September. (2002)
4. S.Jang, and W.Woo, "ubi-UCAM: A Unified Context-Aware Application Model", Lecture Note Artificial Intelligence (*Context'03*), Vol, 2680, pp.178-189, 2003
5. Y.Oh, W.Woo, "A unified Application Service Model for ubiHome by Exploiting Intelligent Context-Awareness," *Proc. Of Second Intern. Symp. On Ubiquitous Computing systems (UCS2004),* pp. 117-122,2004.
6. S.Jung, W.Woo, " UbiTrack: Infrared-based user Tracking System for indoor environment," *ICAT'04*, 1, paper 1, pp. 181-184. (2004)
7. Garcia-Molina, H. Elections in Distributed Computer Systems. *IEEE Transactions on Computers*, Vol, C-31, No. 1, pp. 48-59. (1982)

Choonsung shin received the B.S degree in Computer Science from Soongsil University in 2004. Now he is a M.S. student in Department of Information and Communications, Gwangju Institute of Science and Technology (GIST) since 2004.

Research Interest: Context Awareness, Human Computer Interaction, Ubiquitous/ Wearable Computing.

Woontack Woo received his B.S. degree in EE from Kyungpook National University, Daegu, Korea, in 1989 and M.S. degree in EE from POSTECH, Pohang, Korea, in 1991. He received his Ph. D. in EE-Systems from University of Southern California, Los Angeles, USA. During 1999-2001, as an invited researcher, he worked for ATR, Kyoto, Japan. In 2001, as an Assistant Professor, he joined Gwangju Institute of Science and Technology (GIST), Gwangju, Korea and now at GIST he is leading U-VR Lab.

Research Interest: 3D computer vision and its applications including attentive AR and mediated reality, HCI, affective sensing and context-aware for ubiquitous computing, etc.