History-based Conflict Management for Multi-users and Multi-services*

Choonsung Shin, Yoosoo Oh and Woontack Woo

GIST U-VR Lab.
Gwangju 500-712, S.Korea
{cshin,yoh,wwoo}@gist.ac.kr

Abstract. Context management in context-aware applications manages contexts obtained from various sensors and services related to users and environments. However, most of the previous context management mainly focuses on single user or single application. In this paper, we propose Context Manager that resolves conflicts among multiple users and multiple services for context-aware application in smart home environments. In such environments, the proposed Context Manager resolves conflicts among users by assigning priority to each user based on their context. In addition, it adjusts weights of the context by applying Bayesian theory to conflict history of users and applications. Furthermore, Context Manager detects and resolves conflicts among services by utilizing preferences of users and properties of the services. During experiments on ubiHome, a smart home test-bed, the proposed Context Manager resolved conflicts among users more accurately than resolution method having fixed-priority. The Conflict Manager also resolved conflicts among ubiServices. We expect the proposed method can play a vital role in context-aware applications for offering personalized services to users by resolving service conflicts among applications as well as users.

1 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 [1]. Especially context-aware applications offer appropriate services to users by utilizing contextual information of environment including users. This information is obtained from various sensors or computing resources distributed in our daily life [2] [3]. However, conflicts occur in context-aware applications when multiple users share the applications or these applications share the limited resources in environment [9][10]. A service conflict is a situation in which applications cannot directly provide users with personalized services [19]. The service conflict is classified into three types according to sources of conflicts: service conflicts among multiple users, service conflicts among multiple applications and service conflicts among

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

multiple users and multiple applications. Service conflicts among users are caused due to use of an application by multiple users. Service conflicts among multiple applications are caused by providing of services among multiple applications. Service conflicts among users and applications are caused due to the use of multiple services by multiple users. 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 [4][8]. Reconfigurable Context-Sensitive Middleware for Pervasive Computing (RCSM), an object-based framework, makes sensors and application services independent, forms ad-hoc communication between them, and delivers the necessary context to the applications [5]. Context Toolkit collects, interprets, and delivers context between sensors and application services [6]. Contextual Information Service (CIS) manages contextual information such as location and characteristics of users, devices, and status of network to provide contexts to application services [7].

However, context management techniques in the previous research have 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 [4]. 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 [5]. In the case of Context toolkit and CIS, application developers have to consider both conflict between services and between users because contexts are delivered to applications when current context of environment matches an application-specified condition [6][7].

In this paper, we propose Context Manager to resolve conflicts caused by use of services among multiple users and limited resources among multiple services. In order to resolve the conflicts, the proposed context manager consists of four components: Context Preprocessor, Context Database, Conflict Manager and Final Context Deliverer. Context preprocessor carries out filtering and converting context. Context Database keeps track of several kinds of context. Conflict Manager resolves conflicts among services and among users. Final Context Deliverer sends the conflict-resolved context to applications.

The proposed Context Manager has following advantages. Firstly, it resolves the conflicts among users by assigning priority to users based on their contexts, such as users' identity, time, location and behaviors. Secondly, the Context Manager adjusts weight of context by applying Bayesian probability to conflict history of users. Consequently, it can reflect the change of users' preference and their environment. Thirdly, the Context Manager detects and resolves conflicts among applications by utilizing their properties and relationship between them. Therefore, applications provide users with personalized services by resolving conflicts not only among applications, but also among users.

This paper is composed as follows. In Chapter 2, we introduces unified context-aware application model for ubiquitous computing environments. Chapter 3 describes

the architecture of Context Manager and Chapter 4 explains the conflict resolution method to resolve conflicts among services and among users. Experimental setup and results are discussed in Chapter 5. Finally, we conclude in Chapter 6.

2 A Unified Context-Aware Application Model

In order to provide the personalized service, we represent context information as 5W1H (Who, What, Where, When, How, Why). 5W1H contains comprehensive information about user and his surrounding environment. We defined it as unified context [11]. With each field having sub-fields, the unified context also represents detailed information a user. The unified context expressed with 5W1H ensures independence between sensors and services. It also has advantage of being re-used by other services. In addition, the unified-context ensures reducing additional management required to change the context to other forms according to individual service. ubi-UCAM 2.0 (Unified Context-aware Application Model for ubiquitous computing environment) is context-based application model to provide users the personalized service by exploiting context in ubiquitous computing environments where many different kinds of sensors and services are distributed [20]. Ubi-UCAM is composed of ubiSensors and ubiServices. The ubiSensors and ubiServices exchange contextual information with several types of contexts based on the unified context. Figure 1 shows the overall architecture of the ubi-UCAM 2.0.

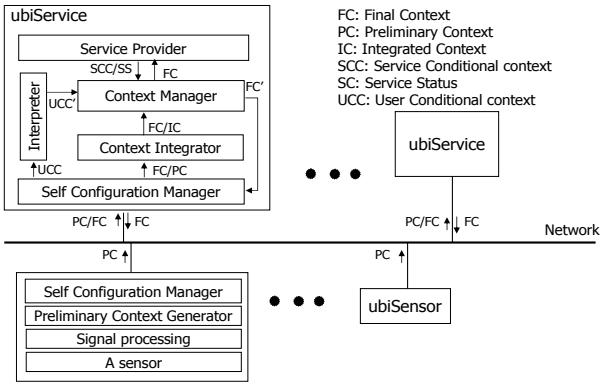


Fig. 1. The architecture of ubi-UCAM 2.0. It consists of ubiSensor and ubiService.

As shown in Figure 1, ubi-UCAM 2.0 employs different types of contexts according to the role of each context. These include Preliminary Context, Integrated Context, User Conditional Context, Service Conditional Context, and Final Context.

Preliminary Context (PC): A unified context which describes current situation of a user and his environment, and includes all or part of 5W1H. It is generated by ubiSensors

Integrated Context (IC): A unified context which describes current situation of a user and his/her environment, and includes all of the 5W1H. It is generated by Context Integrator

User Conditional Context (UCC): A unified context which expresses an action and parameters of a service and related user condition. It is used for generating User Conditional Context'. User conditional Context' is generated by Profile Manger.

User Conditional Context' (UCC'): A unified context which expresses an action and parameters of a service and related user condition. It is used for matching Integrated Context. User conditional Context' is generated by Interpreter.

Service Conditional Context (SCC): A unified context which expresses an action and parameters of a service and related user condition. It is used for matching Integrated Context. Service conditional Context is generated by Service provider.

Final Context (FC): A unified context which describes a user, his environments, and service action and parameters. It is used for triggering a service. It is generated by Context Manager.

An ubiSensor is composed of a physical sensor, Signal Processing module, Preliminary Context Generation module and Self Configuration Manager. The physical sensor perceives a change related to a user and his environment. Signal processing module extracts feature information from the sensed signal. Preliminary Context Generation module generates a preliminary context from the feature information. The ubiSensor delivers this context to ubiServices located within a working area through a multicast group established by Self Configuration Manager. A ubiService is composed of Context Integrator, Context Manager, Interpreter and Service Provider. Context Integrator collects preliminary contexts created by various ubiSensors located within a working area during given time interval. It classifies the preliminary contexts to each sub-element and analyzes the sub-elements by applying a decision making technique. Context Integrator generates an integrated context of each user and delivers integrated contexts based on Context Manager. Context Manager searches conditional context from a Hash-table, which manages specific service action and condition, corresponding for each integrated context. It generates a final context to be used by applications after resolving conflicts among users and services. Finally, Service Provider executes appropriate action with parameters described in the final context. This utilizes application-specified methods which are programmed by application developers.

3 Context Manager

In ubiquitous computing environments, computers with limited memory and processing power are embedded in objects or appliances. In the limited and distributed computing environments, Context Manager maintains only small amount of information, unlike the centrally managed systems that keep all the information [4][7]. Considering the limitations, the proposed Context Manager mainly focuses on the ability to resolve conflicts cause by multi-user and multi-devices in context-aware applications.

In addition, short-term memory in Context Manager manages the history of users and services to resolve conflicts dynamically. To deal with the conflicts in a dynamic way, Context Manager preprocesses context, resolve conflicts with history management, and generates final context. Figure 2 illustrates an overall architecture of the Context Manager.

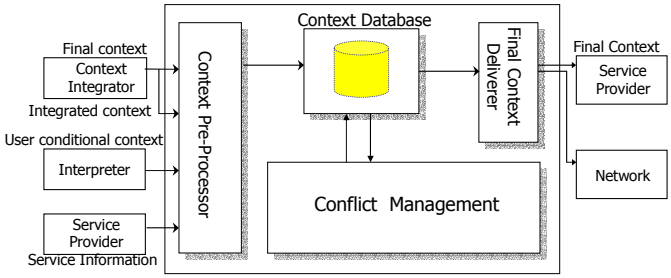


Fig. 2. Context Manager. It consists of Context Preprocessor, Conflict Manager, Final Context Deliverer, and Context Database

As shown in Figure 2, Context Preprocessor receives final contexts and integrated contexts from Context Integrator. It also receives conditional contexts from Interpreter and Service Provider. Then, it stores the resulting contexts to Context Database. The Context Database keeps track of the contexts and maintains conflict history of users and services to resolve conflicts among users and services. Final Context Deliverer builds a final context after resolving conflicts among users and services, and then delivers it to Service Provider and Network module.

3.1 Context Preprocessor

Context Preprocessor carries out pre-treatment of several kinds of contexts, such as integrated context, conditional context, and final context. Figure 3 shows context processing in Context Preprocessor.

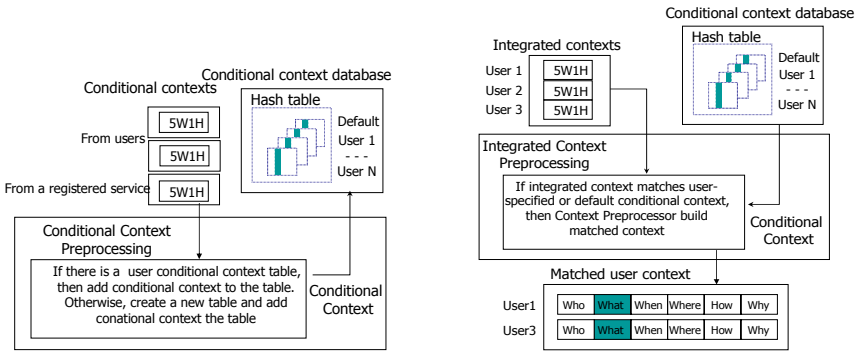


Fig. 3. Context Preprocessing

As shown in Figure 3, Context Preprocessor inserts user conditional contexts and service conditional contexts to Hash table. In the case of integrated contexts, Context Preprocessor yields a matched user context after matching integrated context to conditional context. Context matching is archived in two ways: by matching integrated context with user-specified conditional context and matching it with default conditional context. In each step, if integrated context matches a conditional context, Context Preprocessor builds a matched user context by combining 'What' field of the conditional context and 'Who, When, Where, How and Why' fields of the integrated context. Furthermore, Context Preprocessor collects and filters the final contexts from other services within working area.

3.2 Context Database

Context Database keeps necessary contexts and related information in order to support generation of final context. It contains several kinds of context such as conditional context, final and user context within a service area, and conflict history of users and a registered service. Conditional context table stores conditional contexts generated by users and a service developer. User context table keeps context of the users who are currently related to a service. The user context table is used to select one user when users leave the service area. The final context table storing final contexts of other services is used to confirm whether conflicts are caused within the service area.

3.3 Final Context Deliverer

Final Context Deliverer offers the final context, which has service action and information about its user, to Service Provider and Network module. Therefore, Final Context Deliverer performs two kinds of works: context generation and context delivery. In context generation, it makes a final context which has unique information of the service and doesn't cause conflict. Next, Final Context Deliverer provides the context with Service Provider. It also confirms whether the context is reflected to the service by utilizing service status information coming from Service Provider. With the result, Final Context Deliverer notifies the change of the service to other services if the result is true. Otherwise, it tries to again send the final context, if the service did not work on the final context.

4 Conflict Management

Conflicts of context-aware applications occur not only due to multiple users who access services at the same time, but also due to services trying to share resources in their surrounding. To solve conflict among users, Conflict Manager assigns priority to users and chooses the user given the highest priority. In addition, to deal with conflict among services, Conflict Manager detects and resolves conflicts, based on the

properties of services and relationship between them. Meanwhile, priority of users and services are not fixed, but adapts to user's preference and behaviors. Therefore, Conflict Manager not only resolves conflicts among users and among services, but also dynamically assigns priority to users and services.

4.1 User Conflict Manager

User conflict Manager resolves conflicts caused by users who try to use services within a service area. To resolve the conflict, User Conflict Manager manipulates user contexts in two steps: building a user conflict list and selecting a proper user from it. 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 a remote controller or Tangible Media Controller (TMC) [13]. In the next stage, User Conflict Manager chooses one user from the conflict list based on user's priority according to context weight to user's context calculated from Conflict History Manager. 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 is more than one user within a service area. In this situation, User Conflict Manager selects the user having the highest priority because conflicts may occur. In addition, it notifies the result of conflict resolution to enable Conflict History Manager to store conflict context. Figure 4 shows an example of service conflict among users and a resolution procedure on it.

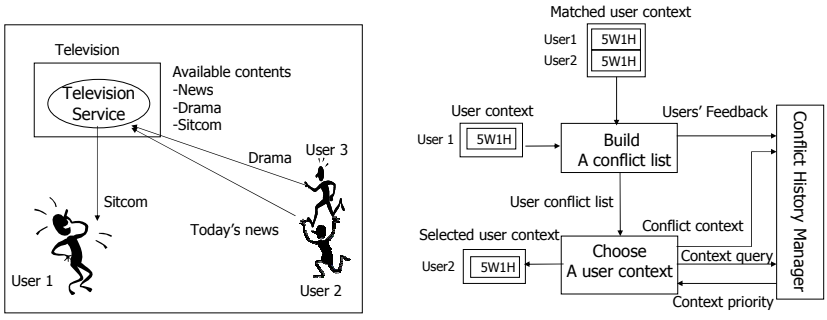


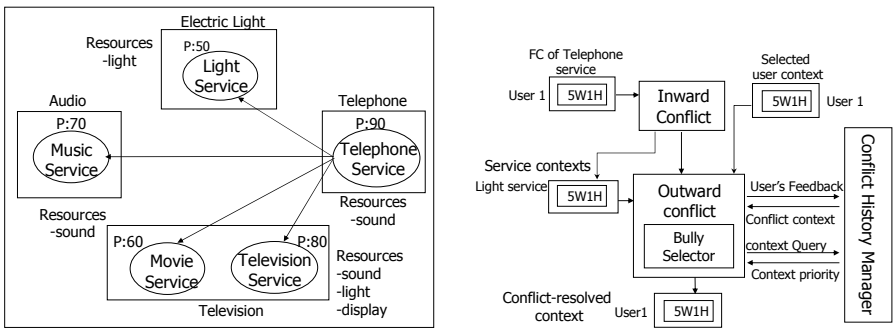
Fig. 4. A service conflict among users (a) and resolution procedure (b)

As shown in Figure 4(a), there is a television service providing user 1 with sitcom program in a service area. Simultaneously two users, user 2 and user 3, are trying to

use the service. In this scenario, we assume that user 2 usually uses the service when conflicts cause among them. Therefore, a service conflict arises due to use of television services by the three users. In this conflict situation, User Conflict Manager, shown in Figure 4(b), detects the conflict and builds a conflict list consisting of contexts of user 1, user 2 and user 3. Based on the conflict list, it then queries priority of each user from Conflict History Manager. User Conflict Manager obtains the priority of users within the service area. Finally, it selects user 2 having the highest priority.

4.2 Service Conflict Manager

Service Conflict Manager resolves conflicts caused by multiple ubiServices trying to share resource in a service area. It deals with conflict in two ways: inward conflict resolution and outward conflict resolution. In inward conflict resolution, it resolves conflicts caused by other services within a service area. Service Conflict Manager creates a context which contains information about the service and a stop command for it, if resources involved in other services are the same as those of the service itself. As a result, the application responds to changes of other services which cause conflict, using final contexts coming from other services. In case of outward conflict resolution, Service Conflict Manager prevents the registered service causing conflict with other services. 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 Deliverer 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. Figure 5 shows a conflict situation among ubiServices and a resolution procedure on it.



(a) (b)

Fig. 5. A service conflict among ubiServices (a) and resolution procedure (b)

As shown in Figure 5, there are four devices: eclectic light, audio player, television and telephone. Audio provides music service, electric light provides light service, telephone provides telephone service and television provides movie and television

services. These services also utilize specific resources of each device. In this situation, the telephone service causes a conflict with television service due to the sound resource. Therefore, Service Conflict Manager detects the conflict in inward conflict resolution. It then builds a final context containing a stop command for the registered service. Afterward, Service Conflict Manager compares the priority of the registered service with the priority of telephone service. According to the priority, it selects the context of the telephone service.

Service Conflict Manager also deals with the situation when multiple services want to use resources at the same time. This is because services can respond to the same condition. In the case of this conflict, several services 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 with the highest priority [18]. In case of service conflicts, the algorithm is used to choose the ubiService having the highest priority among ubiServices which try to use shared resources.

4.3 Conflict History Manager

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 [12][14][15]. Figure 6 shows the overall architecture of Conflict History Manager.

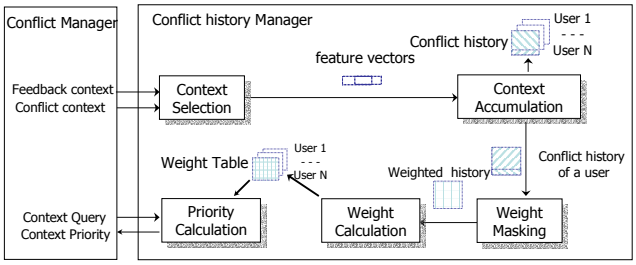


Fig. 6. Conflict History Manager

As shown in Figure 6, Conflict History Manager receives feedbacks and conflicting contexts of users from Conflict Manager. Based on the contexts, Conflict History Manager builds feature vectors containing information about the conflict situation. Each feature vector is represented in the Table 1. Afterwards, the feature vector is stored in a history file so that it can be retrieved when required. Then, Conflict History Manager loads the feature vectors, related to a specific user, from conflict history. Conflict History Manager manipulates weights of conflicting contexts used for priority calculation of users. Therefore, it recalculates weights of conflicting contexts

based on feature vectors of a user. In order to obtain the weight, Conflict History Manager applies Bayesian theory to the feature vectors.

Table 1. The features of feature vectors

Feature		Category
What	Service[SVC]	Content 1: 0 Content 2: 1 Content 3: 2 Content 4: 3
Where	Location[L]	Other: 0 Front: 1 Side: 2 Entrance: 3
When	Time[T]	Hour: 0~23
How	Gesture[G]	None: 0 In: 1 Out: 2 Sitdown:3 Standup: 4 Moving: 5
Who(other)	Who_conflict[WC]	ID
Target class	Resolution result	Selected: 1 Not Selected: 2

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 SVC (Service Type), L(location), T(Time), G(Gesture), S(Stress), and WC(Who_conflict) in table 1. The result of conflict resolution H_j, which is represented by (H_1, H_2) indicates the SC which is 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 current user of a service 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, 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, 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 also 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 current user will continue using the service, and posteriori probability as $P(X_i|H_2)$ when other 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 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 two modules, Conflict History Manager adjusts the weight of conflicting context by 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

5 Implementation and Experiments

We have evaluated the effectiveness of the conflict resolution method based on the ubiHome test-bed. The proposed conflict resolution method chooses one among several users when multiple users attempt to access a registered service. In addition, it decides to start the service when priority of the service is higher than other services located within a service area. We also compared accuracy of the proposed method with fixed priority method in each service. Furthermore, we surveyed on how much of the family members conflict with each other in daily activities. Finally, we conducted a survey on the usefulness of the conflict resolution method to family members.

5.1 Experimental setup

The proposed Context Manager was implemented with J2SDK 1.4™ so that it can be applied to various applications. As shown in Figure 7, we tested Context Manager in ubiHome, a smart home test-bed [16]. In ubiHome, we utilized various ubiServices such as, television service, music service, movie service, light services, etc, which offer customized services to users. In addition to the services, we also exploited various sensors: ubiCouch sensors, ubiTrack, TMC (tangible media controller) and ubi-

Remocon. The ubiCouch sensors are couch sensors, comprising of on/off switches and PIC16F84, detect user's behaviors. ubiTrack is infrared-based location tracking system that tracks user's location [17]. TMC is a tangible media control object to manipulate these services [13]. ubiRemocons are remote controllers, based on Personal Java, to control these services [21].

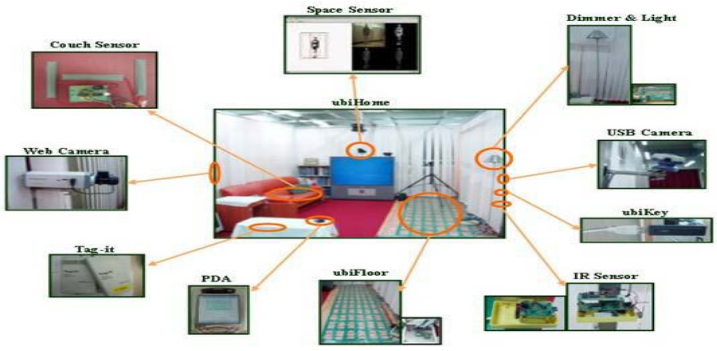


Fig. 7. ubiHome test-bed

To set up the condition of each ubiService, we conducted a survey on service preferences of users about their home environment. The survey was conducted for the home appliances frequently used in living room. Seventy persons, (40 parents / 30 children), were asked the following questions.

Question 1: What kind of services or appliances do you use frequently in your home?
Question 2: When do you usually use the services answered in the Question 1?

As a result, we found that parents usually spend their time watching television around 9 P.M. Especially, they prefer to watch news to get social or weather information through the television. However, children usually consume their time by watching movie and listening to music. They also enjoy watching sitcom or comic programs through television. Based on their preference and time, we assigned conditional context of each user to each service in ubiHome. Furthermore, for the experiment, the number of members in family was four: two parents and two children. This is the average number of members in Korean family system [22].

5.2 Experimental analysis

In order to measure the accuracy of resolution method of the proposed Context Manager, we experimented on user conflict in two ways: a resolution method based on the Bayesian theory and a resolution methods having fixed priority. The proposed resolution method assigns priority to conflict contexts and then chooses one user having the highest priority when conflict occurs. On the other hand, resolution method having fixed priority chooses one user which belongs to the eldest user in the conflict situation. To test two methods, we employed television service that users use most 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 broadcast channel of a user. It decided a specific channel of the user who has the highest priority according to each selection strategy when conflicts occurred. The service gathered feedback of 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. Based on the selection result, we have built confusion matrix to know how well it works. First of all, we tested user conflict with fixed resolution method. We did the experiment from 18:00 to 24:00 in two weeks. Table 2 shows confusion matrix of conflict resolution method using fixed priority.

Table 2. Confusion matrix for conflict resolution having fixed priority (unit :%)

Users	Father	Mother	Son	Daughter
Father	100	0	0	0
Mother	31	69	0	0
Son	43	20	37	0
Daughter	51	29	20	0

As shown in the table 2, the resolution method provided the service to only a specific user because the conflict solution method selects one user according to the fixed priority of users when conflict occurred. Therefore, the higher priority users have, the more services the users have. In the case of father, the rate of correct selection reached to 100% since he has the highest priority among family member. However, the daughter hardly used the television service based on her context due to her lowest priority. Furthermore, the method cannot encompass the changes of preference or behaviors of users in their home environment. This shows how the resolution method is inappropriate to context-aware applications. On the other hand, resolution method, based on the Bayesian theory, reflected these changes. Table 3 shows the experimental results of the proposed conflict resolution method.

Table 3. Confusion matrix for conflict resolution based on Bayesian theory (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 the table 3, the resolution method gave the television service to other users who have lower priority in the conflict resolution having fixed priority. This is because conflict resolution method assigned priorities to users based on their context. In addition, the accuracy of the resolution method was relatively higher than the fixed resolution method. The improvement of accuracy is due to the fact the resolution

method reflected the changes of their preference and resolution policy. Therefore, conflict solution resolved conflicts caused by use of services among multiple users.
In addition, we configured properties of ubiServices to deal with conflict among services. In the experiment, all the ubiServices were in the same area. Especially, television, movie and Internet services were operated on the same computer. Table 4 shows the properties given to each ubiService.

Table 4. Property of ubiServices

Services	Sound	Display	Light	Priority
Television		O	O	90
Audio	O			50
Light			O	60
Movie	O	O	O	80

As shown in the table 4, each ubiService has its own set of required resources, such as sound, display and light, according to the resources it uses. Therefore, services which require the same resource cannot be executed simultaneously. Such service can start after stopping other services. For example, the television service uses sound, display and light resources and Internet service needs display resources. In this situation, those two services cannot be executed at the same time, because they share the display resource. Then, we monitored the services in ubiHome in order to observe resource conflicts among services. Table 5 shows the amount of service conflicts found during the observation.

Table 5. Amount of Service Conflicts in each ubiService (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 Music service. The rest of the conflicts are associated with movie service. Movie service, which shares sound, light, and display resource, is related to all the services. In particular, conflicts with Movie service are mostly due to television service which is accessed by users frequently. Besides, Movie service also conflicts with electric light service since the services use light resource. Music service was related to television and movie service using sound and display resources. Finally, conflicts of the electric light service are caused by movie and television services which share light resource. Therefore, conflicts between these services depend on users and their pattern of using services in home environments.

Finally, we questioned 70 volunteers in ages from 10 to 60 who had experienced context-aware service supporting conflict resolution, in order to estimate the conflict in home environment. They were asked to answer the following questions.

Question 1: Who is the most related to you when you are trying to use television service.

As shown in Table 6, the conflicts appeared high in the viewpoint of parents when they were using the service with their children. In the case of children, they showed high conflict when they spend their time on using the service with other brother or sisters. This result implies that conflicts are occurred because the preferences of each family member are different in using services in home environment. Moreover, each member feels a service conflict differently. This is because the persons who are together are different with each others, when they spent their time on using the services in the living room.

Table 6. The amount of conflict among family members (Unit: %)

	Father	Mother	Children
Father	-	40	60
Mother	30	-	70
Children	33	20	47

Furthermore, to evaluate the effectiveness of the proposed conflict resolution method, we asked them to answer following question.

Question 2: What do you think of context-aware services that choose a proper user when several members try to use them at the same time?

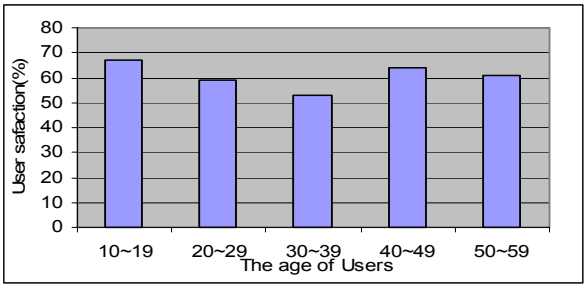


Fig. 8. User satisfaction

As shown in Figure 8, the respondent showed a satisfaction rate to correspond to average 62%. 10s and 40s of the users showed a high satisfaction rate on the survey. Especially, the teenagers showed relatively higher satisfaction then other ages. This is because they have a lot of curiosities with the services, and adhere to use the services. In case of 40s, they showed higher satisfaction rate on using the services too. This is because they want to use their service although there is more than one member. On the other hand, in case of 20s and 30s, lower satisfaction rate appeared. In the case of 30s, their families were comprised of only a few members of family. Most of 20s are live alone. Therefore, we found that satisfaction rate of usefulness of the service resolving conflicts are related to users and their environment.

6 Conclusion

In this paper, we proposed the Context Manager to resolve conflicts that arise when multiple users access various context-aware applications and when the applications are trying to share resources in ubiquitous computing environments. In order to resolve conflicts among users, the proposed Context Manager maintained the conflict history of users, calculated the weight of context with Bayes theory, and then selected one having the highest priority among users. In addition, Context Manager detected conflicts among services based on the resource properties of each service. These conflicts were resolved with the priority so that the services are exclusively executed. Through the experiment, we have shown the effectiveness of the proposed method. In our future works, however, we will employ additional services to deal with the conflicts. We will also observe user's behaviors over longer periods. These remaining works will be done step by step in the near future.

Acknowledgement
The authors would like to thank Seiie Jang for his valuable input and useful discussion about conflicts in context-aware applications

References

1. Mark Weiser. Computer of the 21st Century. Scientific American, 265(3): 94-104, September. (1991)
2. Schilit, B., Adams, N. Want, R. Context-Aware Computing. *Proceding of the 1st International Workshop on Mobile Computing System and Applications*, pp. 85-90. (1994)
3. Anind K. Dey, "Understanding and Using Context. Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing, 5(1),. (2001)
4. Nicholas Hanssens, Ajay Kulkarni, Rattapoom Tuchinda, and Tyler Horton, "Building Agent-Based Intelligent Workspaces," In *ABA Conference Proceedings*, June. (2002)
5. Anind K. Dey and Gregory D. Abowd, The Context Toolkit: Aiding the Development of Context-Aware Applications, *Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing (SEWPC)*, Limerick, Ireland, June 6. (2000)
6. 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)
7. Judd, G, Steenkiste, P., "Providing Contextual Information to Pervasive Computing Applications", *IEEE International Conference on Pervasive Computing (PERCOM)*, Dallas, March 23-25. (2003)
8. John Canny, Danyel fisher, "Active-Based Computing," in *Proceeding of CHI, The Hague, The Netherlands*. (2000)
9. Christian Kray, Rainer Wasinger, and Gerd Kortuem, *Proceedings of the workshop on Multi-User and Ubiquitous User Interfaces (MU3I) at IUI 2004*, Funchal, Madeira, Portugal, ISSN 0944-7822, pp. 7-11. (2004).
10. Meyer, S. and Rakotonirainy, A survey of Research on Context-Aware Home. *Proc. Of the Australasian information serucrity workshop conference on ACSW frontiers 2003*, pp. 159-168. (2003)

11. 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
12 Jiawei Han, Micheline Kamber, *Data Mining: Concepts and Techinques*, Morgan Kaufmanm. (2001)
13. Sj.Oh, W.Woo, “Manupulating multimedia contents with Tangible Media Control”, LNCS(*ICEC*), vol.3166, pp.57-67.(2004)
14. Anand Ranganathan, Jalal Al-Muhtadi, Roy H. Campbell, Reasoning about Uncertain Contexts in Pervasive Computing Environments.. In *IEEE Pervasive Computing,* pp 62-70, Apr-June, (2004)
15. Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Haikki Keranen, and Esko-Juhani Malm, Managing Context Information in Movile Device, In *IEEE Pervasive Computing,* pp. 42-51, July-September, (2003)
16. 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.
17. S.Jung, W.Woo, " UbiTrack: Infrared-based user Tracking System for indoor environment," *ICAT'04*, 1, paper 1, pp. 181-184. (2004)
18. Garcia-Molina, H. Elections in Distributed Computer Systems. *IEEE Transactions on Computers*, Vol, C-31, No. 1, pp. 48-59. (1982)
19 C.Shin and W.Woo, “Conflict Resolution among Users by Utilizing Context History”, *the 3rd International Conference on Pervasive Computing 2005 workshop*, will be published.
20 Y.Oh, C.Shin S.Jang and W.Woo, “ubi-UCAM 2.0: Unified Context-aware Application Model for ubiquitous computing environments”, *the 1st Korea/ Japan Joint workshop on Ubiquitous Computing and Network Systems*, 2005, will be published.
21. http://java.sun.com/products/personaljava/
22. http://www.nsf.or.kr