A Group-aware Service Discovery Scheme in Ubiquitous Environment using Service Assignment

Hankyul You, Dohyun Kim, X.T. Hoang, Younghee Lee
School of Engineering
Information and Communications University
110 Munji-ro, Yuseong-gu, Daejeon, Republic of Korea
{hanky. t12t12t12, tung_hx, yhlee}@icu.ac.kr

Abstract

In this paper, we describe a Group-aware Service Discovery (GASD) scheme for ubiquitous computing. GASD considers the group context in the service discovery for the sake of improving both the system performance and overall user satisfaction. The main features of GASD are as follows: 1) Group Query Aggregation *which collects requesting queries and generate a group query from multiple requests of users to avoid conflicts in the sharable service, 2)* Sharable Service Assignment *which assigns an appropriate service including the private service and the public service to every user based on the group query for the service sharing. Our experimental result shows that GASD has better system performance and overall user satisfactions than the existing semantic service discovery systems. Also, it has better quality of the request result by the efficient assignment algorithm.*

1. Introduction

Nowadays, the rapid development of the computing technology enables users to utilize the mobile devices within the wireless communication network in our daily life. These recent advances have generated a new computing paradigm[1], "ubiquitous computing", in that the different kinds of the devices and the objects coexist and communicate with each other to provide the user with the better service. In that sense, discovering the appropriate service becomes the key requirement in the ubiquitous computing to realize the smart environment for the users. The initial stage of the service discovery starts from the simple syntactic matching where the service itself is described by a single service type or the interface name. Therefore, a user can easily request the service by sending a simple query that exactly matches the service. However, as the scale of the computing environment becomes larger, the number of the service is also proportional to the environment scale that a service can't be distinguished by a simple service type or a name. In this situation, the exact matching with the simple query is no longer useful to provide the most appropriate service. To overcome this constraint, the semantic service discovery[12, 13] is introduced to support alternative services without the exact service information. The semantic service discovery is achieved by the semantic description of the service which is mostly generated by the fine-grained ontology. Moreover, the adaptability of the system is advanced to realize the context-aware service discovery[9-11] which makes use of the contextual information, such as the location, the availability, the current status, and etc., to offer the better service for a requesting user in the case of lots of relevant services having the same type. In this case, the system can also evaluates and ranks the relevant services[7, 8] using the context information and the user preference to recommend the right one to a single user. In spite of such a great enhancement in the service discovery and evaluation, it only deals with a individual user context which lacks the concept of the group-awareness[3] where multiple users interact with each other when they perform their own task with different services. Considering the group-awareness in the service discovery is essential when the scale becomes larger that a larger number of users and services exist in a single domain. In the real situation, it is possible that multiple users request similar services with different purposes so that the conflict may happen when the same service is assigned to them. Therefore, recent researches[5, 6] in the service discovery concern intensions of multiple users for conflict-free in the service utilization. However, they only focus on resolving conflicts according to the group context while they didn't grasp the common behaviors that lead to sharing a service for the same purpose. For this, we propose a novel group-aware service discovery scheme, called GASD, for the conflict resolution as well

978-1-4244-3367-4/08/$25.00 ©2008 IEEE

as the sharing among multi-users in the public domain of the ubiquitous environment.

The main goal of GASD is to maximize overall user satisfactions and service utilizations by ensuring relationships among multiple users within the service during the discovery process. To satisfy the main goal, GASD is mainly divided into two parts. First, we introduce the group query aggregation to gather the group contextual information to avoid the conflict within the same group. Second, we show an efficient service assignment scheme for the collaboration of multi-users in a public service. We expect that GASD outperforms existing service discovery systems with increasing number of users and services. In addition, the requesting user can use the better quality of the service without causing the conflict so that the overall user satisfaction can be maximized.

The paper is organized as follows. The section 2 summaries the related work. The design consideration for our proposed scheme is described in section 3. And, we show our proposed scheme in the section 4. We evaluate our proposed scheme in section 5. Finally, we conclude our paper in section 6.

2. Related Work

There are lots of research efforts on the service discovery for the better service utilization and the support of more relevant services. Some efforts turned out to utilize the surrounding environments into the system for the more intelligent discovery. In order to allow the service discovery system to adapt to the current environmental status[9-11], they use the contextual information to provide the most appropriate service to the user in the case of having a number of relevant services with the same type. For example, when a user requests the "printer" service, the system provides the "printer" service which is closest to a requesting user among all the existing "printer" services. In this case, the location is used as a context for providing the better quality of the service. Some other contexts such as the load of the printer, availability, and the amount of remaining inks can be utilized in the service discovery for the better suggestion. However, these service discovery systems are based on the single user semantic that the system handles only one user at a time. And also, it doesn't consider the group-awareness[3] where many users are interacting with each other that the conflict may occur due to these interactions. Under this condition, the system can't guarantee the availability of the returned service for a user. Eventually, this situation decreases the service utilization and quality of the service. To overcome this, recent service discovery schemes[5, 6] consider interaction of multiple users for the service utilization. They try to resolve conflicts by the elaborate effect description which describes the state changes of the environment when the service operates its own function. However, they only consider the situation of conflict when multiple users compete for a single service, but not the case of the public sharing where multiple users share a single service at the same time.

3. Design Consideration

To enable our system to be smart in ubiquitous environment, it is essential that the system needs to be aware of not only individual context, but also group context to consider relationships between users and services. For this, we focus on tracking group contextual behaviors to consider the conflict as well as the public service sharing for the better service utilization. We define the conflict as a situation when more than two users want to use a same service with different operations while the sharing indicates the similar behavior among different users so that it could be revealed as a public ownership within a same service. Secondly, we consider the general service model which can cover the private service and the public service to enable the system to adapt to ubiquitous environment. Finally, because our service discovery scheme deals with the group query for multiple users, we apply an efficient assignment algorithm to assign the appropriate service to every user for the user satisfaction. Moreover, our assignment algorithm focuses on how to support the N-to-1 matching problem for a sharable service.

4. Proposed Scheme

To enable the group-aware service discovery which considers the group situation for multiple users, our scheme is composed of two-folds: 1) *Group query aggregation* which aggregates requesting queries and generate the group query for the group-awareness, 2) *Sharable service assignment scheme* which assigns the proper service to every single user given multiple users and multiple services. We start from discussing the overall structure and continue to explain the details.

4.1. Overall Structure

Figure 1 shows the overall structure of our system. *Query Aggregator* collects the requesting queries asynchronously from the users and generates the group query. Meanwhile, for each query, the incomplete preference is extracted and delivered to the *User Preference Manager* to generate the complete user preference set. Then, a group query is transferred to *Semantic Service Discovery* component, and it discovers

candidate services for multiple requesting users. *Fitness Calculation Function* component ranks all returned services for group users based on their preferences. Every time it ranks every candidate service, it utilizes the user preference, the contextual information for more accurate ranking. After this stage, *Service Instance Maker* component generates the clone instances for shareable services among candidates to enable 1-to-1 matching for the assignment. The number of instances is determined by the remaining service capacity and its unit follows the number of people. Finally, the system assigns the proper service to each user using the existing assignment algorithm which is solvable in a polynomial time.

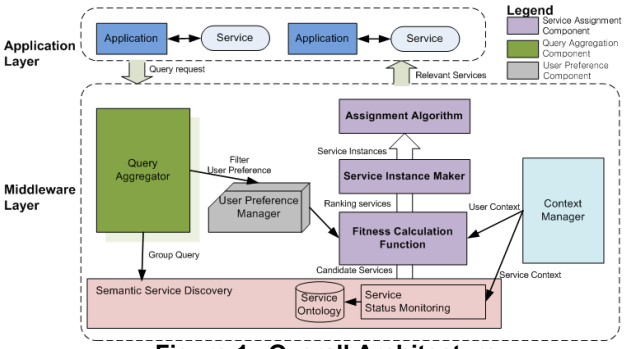

Figure 1. Overall Architecture.

4.2. Group Query Aggregation

To achieve the group-awareness in the system, our proposed scheme aggregates the queries by the *Query Aggregator* to reflect the user interactions during the service discovery process. We assume that there is a reasonable query aggregation interval so that *Query Aggregator* can obtain multiple queries that are enough to be aware of group contextual behaviors without causing the significant query response delay.

Whenever the request query is delivered to this component, *Query Aggregator* extracts its own preference and passes it into the *User Preference Manager*. We assume that *User Preference Manager* is already predefined that it generates a complete set of the user preference which consists of attribute-value(AV) pairs with their weight factors from the incomplete user preference. And this complete user preference is delivered to *Group Query Maker* to be added into the group query. After that, *Group Query Maker* generates the group query from the aggregated queries based on the user intention. Thus, all the queries in the same group require the similar functionality of the service so that the system can avoid the conflict in a sharable service. Then, these group queries are sent to the semantic service discovery synchronously to discover the candidate services. Meanwhile, for every group query, a set of user preferences is forwarded to *Fitness Calculation Function* to be used as a calculation of how much a user is satisfied with the candidate service. *Figure 2* shows the whole process of how user queries are aggregated and transformed into the group queries.

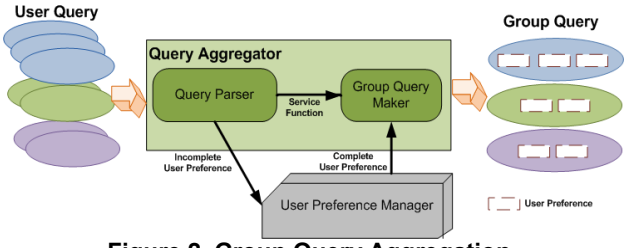

Figure 2. Group Query Aggregation.

4.3. Sharable Service Assignment

Since the semantic service discovery returns the candidate services for a group query, our scheme needs an efficient assignment process for multiple users to maximize overall user satisfactions as well as the public service sharing. For this, we describe a fine-grained service model and the fitness calculation function to measure the fitness value of all candidate services as a preliminary purpose. And then, we apply a sharable service assignment scheme to assign the appropriate service for group users.

4.3.1. Service Model

To reflect the heterogeneity of services in ubiquitous environments, we define our own service ontology model designed by OWL[15]. The structure of our service ontology is represented in *Figure 3*. We assume that a service is a kind of the device having a list of operations in ubiquitous computing, and sometimes it can be shared. To reflect these characteristics, we classify the service into two categories: private and public. A private service can be used by only one user, while a public service can be shared by multiple users concurrently. A service consists of a set of attribute-value(AV) pairs to enable the semantic description of the service with its contextual information. Meanwhile, the semantics of the attribute value is defined in the hierarchical ontology for the calculation of the similarity between two values to be used in *Fitness Calculation Function.* In most cases, by comparing each value of the attribute and the contextual attribute value on the user preference[8], the system estimates the service by the user request and the estimated value is regarded as the user's satisfaction on a service, which we call *fitness value*. For the representation of the service operations, the task with its input and output is added into our service ontology. In addition, a service has only one active operation to show that multiple operations can't be activated concurrently. The service has its own capacity to represent its maximum accumulation of the users for the sharable case.

Formally, it is divided into the overall capacity and the remaining capacity to show how many users are currently using the service out of the maximum capacity.

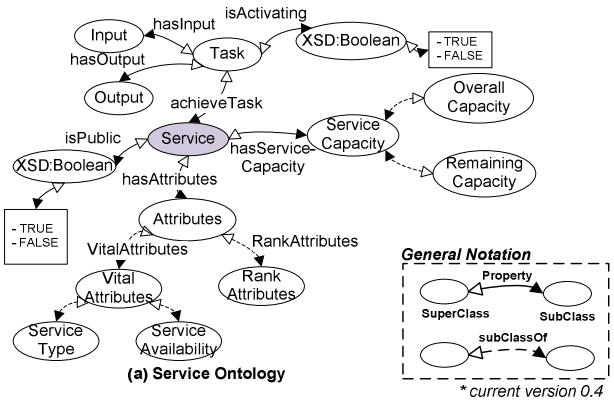

Figure 3. Service Ontology.

4.3.2. Fitness Calculation

In order to solve the assignment problem given *m* users and *n* services, somehow, the fitness value f_{ij} for matching a user i and a service j should be calculated firstly. In this calculation process, *Fitness Calculation Function(FCF)* gets a help from *Context Manager* with informing contextual changes on the services and the users such as availability of the service, location of the user. Therefore, FCF cooperates with this component for the more accurate calculation of the fitness value (f_{ij}) which enables the system to estimate which service is more suitable for the user. *Figure 4* shows the overall process of the calculation of the fitness values for all participating users.

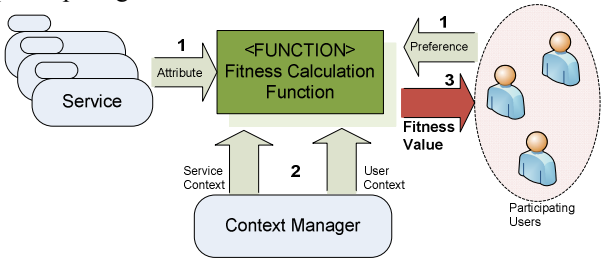

Figure 4. The process of the fitness value calculation.

The fitness calculation function between user preference U_i (i ∈ *I*, *I* : the set of users) and Service S_j(j ∈ *J*, *J* : the set of services) is defined as :

$$F(U_i, S_j) = \frac{1}{n}\sum_{k=1}^{n} w_k \bullet sim(v_{ik}, v_{jk})$$

where n is the number of attributes, v_{ik} and v_{jk} are the values of attribute a_i of user preference U_i and Service S_i, respectively, and w_k indicates the weighting factor of attribute a_i on user preference where $\sum w_i$ is n. We bring the similarity function which measures the similarity between two attribute values from the existing work[14] which shows the best results.

4.3.3. Service Model

After the calculation phase is finished, the system is ready to assign the service to the user using the existing assignment algorithm[4] which is considered as one of the most efficient assignment solutions. However, we cannot directly apply this algorithm into our case because it is based one-to-one matching between the user and the service so that sharable service cannot be applied. To overcome this problem, *Service Instance Maker* component generates service instances for the public services and each generated instance is assigned to at most one user. In addition, each service instance is consistent with the original public service and the maximum number of instances is determined by the remaining capacity of the service so that the generation number does not go to infinity for the system performance. After generating service instances, all the public service instances and private services are not allowed to be assigned to multiple users. That is, this can be regarded as the classical assignment problem[16]. Finally, the system assigns the appropriate service to every single user with the existing assignment algorithm that guarantees maximizing total users' satisfaction. Using this, it is solvable within the polynomial time and it avoids the NP complete problem[16].

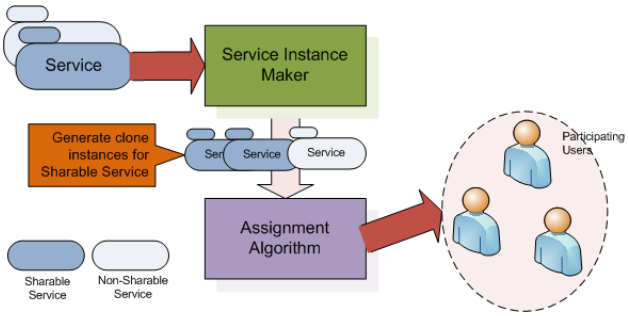

Figure 5. Service Instance Maker and Assignment Algorithm.

5. Performance Evaluation

In this section, we evaluate the performance of GASD and show the experimental results in terms of user satisfaction, accuracy and the system overhead. For this, we implemented GASD running on top of Service Interaction Broker(SIB), which is the remote communication protocol in Active Surroundings middleware[3]. For the experimental purpose, we developed the emulation system which generates fixed number of services and users. For each service, its

attribute values are randomly selected among the predefined set of attributes. An emulated user sends the query for 10 times synchronously during one execution. And its query is generated randomly from the predefined query template and contains the set of attributes with randomly selected values and weighting factors. The requesting user holds the returned service for 5secs when service is not busy. A service is busy when the number of users holding that service is equal to the overall capacity. And, the fitness value can't be obtained with the busy service. Parameter settings for the service are shown in *Table 1* since we believe these settings could be typical in the public space for ubiquitous computing. Finally, we set the query aggregation interval time to 0.5sec showing the best balance in our scheme.

Table 1. Parameter settings for the service.

Parameters	Value
The number of services	20 ~ 50
The number of AV pairs	4 ~ 5
The number of functions for each service	3 ~ 4
The maximum number of users sharing a service	20

To measure the performance of our proposed scheme, we compare our scheme with the existing semantic service discovery which does not make use of the group query aggregation and the assignment scheme. At first, we evaluate the average user satisfaction on the query results. We fix the number of service to 20, and the number of users varies from 30 to 100. We measure the user satisfaction by the average fitness value for all returned services. The result is shown in Figure 6. As you see, there is a great improvement on the average fitness value when applying our scheme into the semantic service discovery. In other words, our scheme provides more relevant services on the user's request because GASD considers the interaction of requesting users within the discovery process.

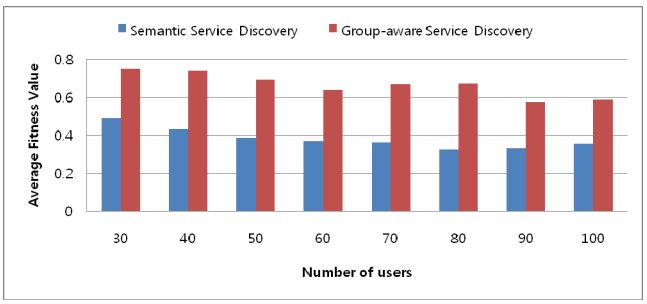

Figure 6. Average fitness value vs. number of users.

Our 2nd experiment is to evaluate the accuracy by the percentage of successful queries which gain the relevant service as a return value. We assume that the query fails when the returned service is none or the returned service is busy. The result is shown in Figure 7. The result shows that the percentage of successful queries of our scheme always larger than semantic service discovery regardless of the number of users that, that is, our scheme supports more accurate results.

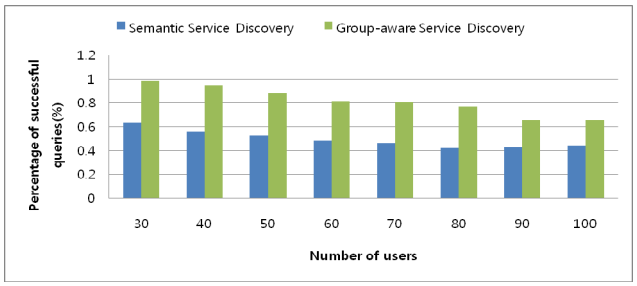

Figure 7. Percentage of successful queries vs. number of users.

Our 3rd experiment focuses on the system overhead by the load reduction with the different number of users and the fixed number of services. Load reduction on the service discovery is measured by the number of group queries sent to the service discovery over the number of all requesting queries. We can see the result in Figure 8. More than 75% of the redundant messages are decreased to access the service discovery, and these rates become larger with the increasing number of users.

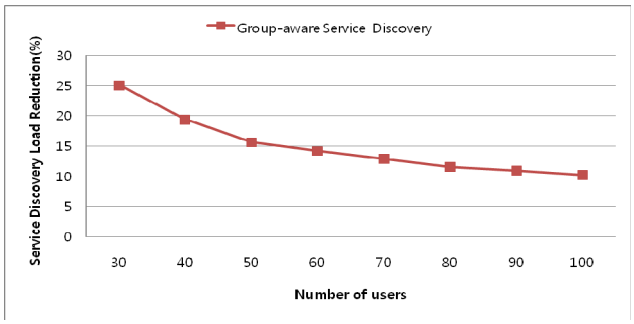

Figure 8. Service discovery load reduction.

From our experimental results, we can conclude that GASD provides the better quality of service to requesting users without the significant system overhead. Moreover, the accuracy of the returned services is guaranteed so that existing services can be fully utilized. However, our scheme needs to wait for the predefined query aggregation interval time that can cause the delay on the return of the result. At the worse case, it can minimize the user satisfaction even though the returned service is very relevant. Therefore, it is very important to decide the optimal query aggregation interval for both user satisfaction and the response delay and we leave it as a future work for the more investigation.

6. Conclusion and Future work

In this paper, we proposed a group-aware service discovery scheme, called GASD, for multiple users and multiple services in ubiquitous computing. The main goal of our scheme is to avoid the conflict in a service as well as the sharing the public service to maximize user satisfactions. For this, GASD has two main findings. First, we construct the new group query aggregation component that enables the system to avoid conflict situation. Second, we adapt an efficient sharable service assignment scheme for the collaboration. As a result of our experiments, from the system's point of view, our scheme reduces the system overhead by reducing redundant requesting messages. From the user's perspective, on the other hand, it provides users with better relevant services improving the satisfactions.

In the near future, we will focus on reducing the query response delay which mostly happens when aggregating the queries for tracking the group context. That's because the delay itself is closely related to the user satisfaction

7. Acknowledgements.

This work was supported by the IT R&D program of MKE/IITA[2007-F-038-02, Fundamental Technologies for the Future Internet].

8. References

[1] M. Weiser, "The Computer for the 21st Century", *Scientific American Special Issue on Communications, Computers, and Networks*, Sep, 1991.

[2] W. Keith Edwards, "Discovery Systems in Ubiquitous Computing", *IEEE Pervasive Computing*, vol. 05, no. 2, pp. 70-77, Apr-Jun, 2006.

[3] D. Lee, S. Han, I. Park, S. Kang, K. Lee, S. Hyun, Y. Lee, and G. Lee, "A Group-Aware Middleware for Ubiquitous Computing Environments", *ICAT 2004*, pp. 291 - 298, Seoul, Korea, Nov-Dec, 2004.

[4] D. P. Bertsekas, "Auction Algorithms", *Encyclopedia of Optimization*, Kluwer, 2001..

[5] G. Kim, D. Kim, X. Hoang, and Y. Lee, "Group-aware Service Discovery using Effect Ontology for Conflict Resolution in Ubiquitous Environment", *ICACT2008*, Feb, 2008.

[6] S. Kang, W. Kim, D. Lee, Y. LEE, "Group Context-aware Service Discovery for Supporting Continuous Service Availability", *ubiPCMM'05,* Sep, 2005.

[7] Abdur-Rahman El-Sayed and James P. Black, "Semantic-Based Context-Aware Service Discovery in Pervasive-Computing Environments", *in Proc. Of IEEE Workshop on Service Integration in Pervasive Environments(SIPE), In conjunction with IEEE International Conference on Pervasive Services(ICPS)*, 2006.

[8] Luke Steller, Shonali Krishnaswamy and Jan Newmarch, "Discovering Relevant Services in Pervasive Environments Using Semantics and Context", *ICEIS Ubiquitous Workshop*, May 2006, Cyprus.

[9] T. Broens, S. Pokraev, M. van Sinderen, J. Koolwaaij, P. D. Costa, "Context-Aware, Ontology-Based Service Discovery", *EUSAI 2004*: 72-83.

[10] Guanling Chen and David Kotz, "Context-sensitive resource discovery", *PerCom 2003*, pages 243-252, Fort Worth, TX, March 2003.

[11] J. E. Bardram, "Activity-Based Service Discovery – An Approach for Service Composition, Orchestration and Context-Aware Service Discovery", CfPC 2004-PB-67, 2004.

[12] Z. Song, Y. Labrou, and R. Masuoka, "Dynamic Service Discovery and Management in Task Computing", *MobiQuitous'04*, 2004.

[13] A. Toninelli, A. Corradi, R. Montanari, "Semantic Discovery for Context-Aware Service Provisioning in Mobile Environments", 9th MCMP 2005, Ayia Napa, Cyprus.

[14] Y. Li, Z.A. Bandar, D. McLean, "An approach for measuring semantic similarity between words using multiple information sources", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 4, pp.871-882, July/August 2008.

[15] OWL. http://www.w3.org/2004/OWL.

[16] Micheal R. Garey and David S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", ISBN: 0716710455, W. H. Freeman Company, November 1990.