Service Assignment Problem: Exploiting Group Context Information in Service Discovery for Multi-user Multi-service Ubiquitous Computing Environments

XuanTung Hoang, Dongman Lee, Younghee Lee
Information and Communications University
119 Munji-ro, Yuseong-gu, Daejeon, Republic of Korea
{tung_hx, dlee, yhlee}@icu.ac.kr

Abstract

We consider in this paper the problem of exploiting group context, context information related to interference and conflict among actions of a group of users, in ubiquitous computing environments. We show that integrating group context to service discovery can be accomplished by solving Service Assignment Problem, a problem that shows requirements of assigning services to requesting users appropriately when there are multiple users requesting for multiple candidate services. A service assignment scheme, named QASA, is proposed as a solution to the problem. It leverages query aggregation to handle the asynchronous nature of query arrival patterns and uses an assignment algorithm to allocate services to users. Our experiments show that QASA improves user satisfaction, service utilization and quality of query results in high load conditions without harming those metrics in low and moderate load conditions. QASA also enhances system performance by reducing redundant accesses to service discovery function.

1. Introduction

Recent advances in mobile handheld devices and wireless communications have motivated a new computing paradigm in that the usage of computing technology involves in everyday activity [14]. That computing paradigm is widely known as ”pervasive computing” or ”ubiquitous computing”. One of key requirements that enable the success of ubiquitous computing is an ability of discovering and selecting appropriate service necessary for user applications. In a typical scenario, a user sends a query to service discovery system to find a service that is required for his client application. The service discovery basically can be seen as a directory server that maintains registered services and resolves service queries to service descriptions.

As pointed by [15], ubiquitous computing environments also pose requirements to service discovery protocols relative to integration with users and their intelligent surroundings. For such requirements, recent service discovery systems for ubiquitous computing environments (e.g. [1, 12, 13]) exploit semantic descriptions of services and context information for query matching and service selection. The semantic service discovery enhances quality of service queries in terms of precision and recall [2]; and the context information, such as location, functional characteristics, current operational states, and etc., is used to evaluate and rank matched services according to their relevancy to the context of requesters. Althoutgh existing context-aware semantic service discovery systems can seamlessly integrate users to the computing environment, those systems ignore *group context*, which relates to interference and conflict between users' actions and only exploit context information that involves *single user semantics*. Those existing works is, thus, valid in an extreme scenario where a single user evaluates a list of candidate services to pick up the best service that his client application requires. They cannot capture complex application scenarios where multiple users compete each other for a common set of services. In reality, the multi-user multi-service scenario exists when there are requests for common services from different users. A typical example of that scenario is a ubiquitous public space like an airport, a supermarket, a park, and etc. In this paper, we consider the problem of exploiting group context in service discovery for multi-user multi-service ubiquitous computing environments. To our best knowledge, our work presented in this paper is the first attempt to leverage group context in service discovery for ubiquitous environments.

The contribution of our paper is two fold. First, we characterize the problem of applying group context to service discovery as *Service Assignment problem* and formulate it as combinatorial optimization problems. We show that the service assignment problem is the conventional *assignment problem* [5] if services are non-shareable, and the problem becomes NP-hard if services are shareable. Second, we propose and describe the implementation of a service assign-

978-0-7695-3427-5/08 $25.00 © 2008 IEEE
DOI 10.1109/UMC.2008.39

ment scheme to solve the service assignment problem for the case of non-shareable service. Our non-shareable service assignment scheme, called QASA, uses a query queue to accumulate and aggregate user queries. Queries that result in the same set of matched services are grouped together. Auction algorithm [3, 4] is used to appropriately assign services to requesters. Our QASA scheme can be seen as a service discovery component that utilizes context information about user interactions to improve results of service discovery. As shown by our experiments, there are several advantages of our proposed scheme over single-user service discovery ones. Firstly, requesters are always assigned the "best services" with an objective of maximizing "social benefit". Effects of this include higher average user satisfaction and better candidate service utilization than service discovery schemes that do not consider service selection in multi-user multi-service situations. Secondly, our proposed scheme improves the quality of query results. Thirdly, through query aggregation, the number of requests to semantic service discovery is much reduced as shown in our experiments. Since semantic service discovery usually involves accesses to a common knowledge base as well as inference on that knowledge base, reducing the number of requests to that heavy-loaded task greatly improves the overall system performance.

The paper is organized as follows. The section 2 summaries related work. The service model and Service Assignment problems are presented in section 3. We describe the design and implementation of our service assignment scheme for non-shareable services in section 4. Section 5 is for performance evaluation. Finally, we conclude our paper in section 6.

2. Related Work

Existing approaches on service ranking [1, 12] focus on scoring service attributes based on their semantic meanings and their relationship with user preferences. In [12], authors combine semantics description of services with context information to improve recall and precision of service discovery. They divide service attributes in service query into vital and rank types. Vital attributes are used to filter out irrelevant services while rank attributes are used to scoring relevant service based on user preference. The system in [1] applies almost the same approach for ranking services by, first, discovering queried services based on static service attributes, which are permanently associated with each service, and then scoring services using dynamic attributes, whose values are context dependent. Although the works solve the problem of converting various semantic and context-aware information of service description into comparable quantities, they just consider a simple situation where only one user ranks a list of relevant services.

The group interaction issues in ubiquitous computing environments are also mentioned as conflict resolution problem. Research efforts on conflict resolution (e.g. [11, 9, 10]) consider interaction among users and services when multiple users compete for a single service. Each service is delivered to a user with a preferred execution rule or policy ([9, 10]). Execution rules of a service to different users may be different and usually conflict to each other. A conflict resolution module determines a compromised execution rule for requesting users in order to tolerate conflicts. An alternative solution, as described in [11], uses priority associated with each requester to resolve conflict. When there are multiple requesters requesting for a single service, the requester with highest priority wins. In order to assign priority to each requester, context history is used. Regarding the scenarios where interactions between users occur, research works on conflict resolution is only a special case of our service assignment problem since they only consider multi-user single-service scenarios while service assignment problem involves multi-user multi-service ones.

3. Service Assignment Problem

3.1. Service Model

We classify services into two categories: non-shareable and shareable. A non-shareable service allows only one user to access at a time while shareable service allows multiple concurrent users. Each service has a set of operational states. The concept of service's operational state is equivalent to the execution rules or policies in [9, 10]. When a service is used by users, one of its operational states is active. Since a non-shareable service is used by a single at a time, we can assume that a non-shareable service has only one operational state. A shareable service, on the other hand, can have several different operational states.

Each operational state is characterized by a feature vector whose elements are contextual attributes associated with a service's state. To illustrate a service operational state and its feature vector, suppose that a TV is currenly showing a certain channel with some volume level. The service operational state of a TV is characterized by a feature vector consisting of two elements: the playing channel and the volume level. Each user associates each contextual attribute with an evaluation function f mapping each possible value of the attribute to a value based on the user's preference which we call a satisfaction value. The feature vector and their evaluation functions are application dependent. A specific application includes only attributes of interest into feature vectors and uses a suitable evaluation function for each interested attribute. A simple form of a user's evaluation function is listing all preferred context attribute values together with weight factors in his service queries. From the

concepts of feature vectors and contextual attribute evaluation function, the utility of a user to a service operation state can be calculated for ranking and evaluation purposes.

3.2. Assignment for Non-Shareable Services

Since a non-shareable service accepts only one user at a time and has a single operational state, we can associate a utility $b_{ij} > 0$ to each pair of a user $i \in I$ and a service $j \in J$. Here I and J are sets of users and services respectively. The objective of service assignment problem for the case of non-shareable service is finding a matching between user set I and service set J that maximizes total utility subject to the following constraints:

1. Each user $i \in I$ chooses at most one service $j \in J$
2. Each service $j \in J$ accepts at most one user $i \in I$

This problem is the classical assignment problem [5] and is polynomial time solvable with various algorithms [3, 4, 8].

3.3. Assignment for Shareable Services

In the situation where there are multiple shareable services, each user chooses at most a service and each service will choose at most one state among its possible operational states. If operational states of services is represented by a set K, for each triple (i, j, k), where $i \in I$ is a requesting user, $j \in J$ is a candidate service, and $k \in K$ is a service state, there is a utility $b_{ijk} > 0$, calculated from the service model described above. Note that we assume every service has a fixed number of states $|K|$. If a service j has less than $|K|$ operational states, we can introduce "dummy states" and associate satisfaction values 0 to all triples (i, j, k) containing those states. The objective of service assignment problem for the case of shareable service is finding a set M of triples (i, j, k) that maximizes the total utility and satisfies the following constraints:

1. Each user $i \in I$ chooses at most one service $j \in J$
2. Each service $j \in J$ chooses at most one service $k \in K$

This combinatorial problem is NP-complete since it can be transformed from Boolean satisfiability problem (SAT) [5] in polynomial time as follows: Given an instance of the SAT problem with a set of clauses C and a set of Boolean variables X, we will construct an instance of the shareable service assignment problem from that instance of the SAT problem. For each clause $c \in C$, we create a user $i \in I$; and for each variable $x \in X$, we create a service $j \in J$. The set K of service states in our shareable service assignment instance contains only 2 operational states T and F. That a service j chooses T or F state is like setting a variable x to $TRUE/FALSE$ value. A triple (i, j, T), $i \in I$, $j \in J$, has utility 1 if and only if in the corresponding triple $(c, x, TRUE)$, $c \in C$, $x \in X$, the clause c contains a literal x, otherwise (i, j, T) has utility 0. Similarly, a triple (i, j, F) has utility 1 if and only if in the corresponding triple $(c, x, FALSE)$, the clause c contains a literal $\bar{x}$ (not x), otherwise (i, j, F) has utility 0. Since each variable can only be assigned either $TRUE$ or $FALSE$, and a clause is satisfiable if and only if at least one literal is $TRUE$, the maximum total utility of the instance of the shareable service assignment problem is reached if and only if all clauses in the SAT instance are satisfiable.

4. Service Assignment Scheme for Non-Sharable Services

In this section, we introduce a non-shareable service assignment scheme, called QASA (Query Aggregation Service Assignment scheme) that solves the service assignment problem for the case of non-shareable services. The solution for shareable service case is under investigation and will be added in near future. For solving the non-shareable service assignment problem, any algorithm that solves the classical assignment problem can be used. In our implementation, we use the auction algorithm. It is considered one of the most efficient algorithms for the classical assignment problem. We refer interested readers to [4, 3] for details of the algorithm.

Since queries are issued by users asynchronously, user set I and service set J in non-shareable service assignment problem cannot be assumed known. The auction algorithm, thus, cannot be directly applied in real ubiquitous computing systems. In order to obtain the sets I and J, we utilize a query queue to accumulate and aggregate user queries. The name QASA of our scheme follows the idea of using query aggregation and service assignment algorithm for ubiquitous applications. Service queries sent by users are put in to the query queue for an interval before being served. During a query queuing interval, queries that result in the same set of matched services are grouped into an aggregated query. Only aggregated queries are sent to semantic service discovery for looking up relevant candidate services. The set of queries that is grouped into an aggregated query represents the user set I, and the set of candidate services returned by service discovery corresponding to the aggregated query is used as service set J. Thus, an instance of non-shareable service assignment problem is determined from a given aggregated query. We say that an aggregated query defines an *group interaction context*. An algorithm that solves the service assignment problem for non-shareable service case will guarantee two characteristics of an *group interaction context*: (1) Conflict-free: All returned services to user queries

are distinct. And (2) utility maximization: requesting users are assigned services in such a way that the overall user satisfaction is maximized. The two characteristics imply optimality of service assignment in terms of conflict resolution and candidate service utilization. Query aggregation is also beneficial to system performance since it reduces redundant access to semantic service discovery module. Semantic service discovery are usually slow since it involves accessing to and reasoning on a common knowledge-base. Reduction in load on service discovery, thus, leads to improvement of system performance.

To implement user preference for service contextual attributes, we develop an XML-based query format that explicitly carries preferred context attribute values together with their associated weight factors. A service query comprises of two parts. One part contains all required attributes and the other lists up optional ones with their weight factors. The part that contains optional attributes and weight factors, in fact, encodes evaluation functions of optional service attributes. Queries that share the same list of required attributes should be grouped into a single aggregated service query by query aggregation.

Although conflict-free interaction context can be guaranteed, service queries belonging to different *group interaction contexts* can conflict each other. There is a trade-off between service discovery delay and the total user satisfaction. A long queue interval will extend group interaction contexts and reduce the number conflict queries between different queue intervals, but it also increases service discovery delay. On the contrary, reducing service queuing interval to shorten service discovery delay will result in more query conflicts and lower overall utility of users. In the extreme case when the queuing interval is zero, service assignment scheme behaves the same as simple service evaluation scheme like [1, 12].

We successfully implemented QASA scheme and integrated it into Active Surroundings middle ware [7]. Active Surroundings middle-ware supports dynamic adaptation and provides a unified communications method between applications and services. It also uses an OWL-based semantic service discovery for service lookup based on service categories, service effects and service methods (including method inputs and outputs).

5. Performance Evaluation

In this section, we present our initial experiments on the performance evaluation of our service assignment scheme in terms of system performance and user satisfaction. For this, we develop a number of emulated user applications that send queries for emulated services. Both emulated user applications and services use Service Interaction Broker, the dynamic re-configurable interaction channel of Active Surroundings middleware, for communications purposes. Ten service templates are used to generate service randomly. We fix the number services to 20 and 50 in our experiments since we believe that those setting could be typical in public space ubiquitous computing scenarios. User applications also generate queries randomly from the predefined service templates mentioned above. Each query containing 5 optional attributes and associated weight factors which are random number between 0 and 1. A user application repeatedly sends 10 queries for services before shutting down. The average rate of queries from a user is about one query per second. When a service is returned as result of a query, if the service is free, the user application that sends the query receives a utility value specified by weight factors in his query and holds the service for a random period. If the service is held by another user, no utility is gained. The average service holding time of a user application is set to 3 seconds.

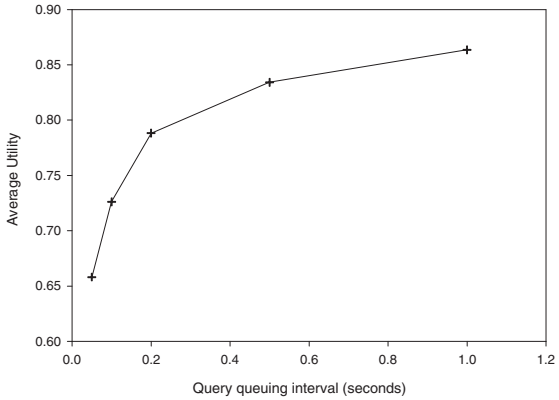

Figure 1. Service discovery load reduction vs. Query Queuing Interval

Our first set of experiments is for investigating the effect of query queuing interval on user satisfaction and on load on semantic service discovery. The results of these experiments are shown in figure 1 and 2. In this set of experiments, there are 50 user applications requesting 20 randomly generated services as described above. The level of user satisfaction is measured by average utility over the total number of queries. The higher average utility reflects a higher level of user satisfaction. As expected, longer query queuing intervals result in higher user satisfaction and less load on semantic service lookup. The reason simply is that more queries can be aggregated within a longer queuing interval. The increment rates of user satisfaction and service discovery load reduction decrease as query queuing interval increases. It implies that there is some optimal value of query queuing interval that balances the trade-off between

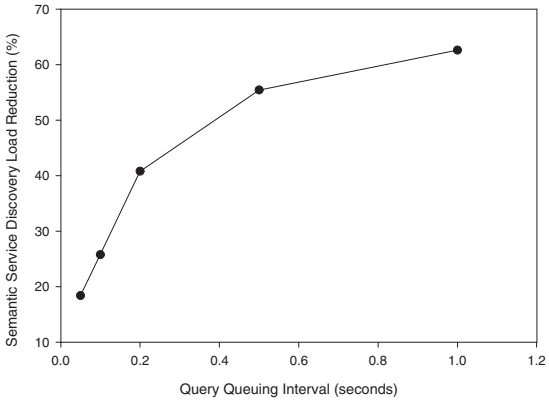

Figure 2. Average Utility vs. Query Queuing Interval

service discovery delay and total user satisfaction. For example, for our experiments settings, query queuing interval = 0.5s could be a best choice since doubling it results in just a little gain in user satisfaction and service discovery reduction. Finding the optimal value of query queuing interval in various scenarios definitely requires further and deeper investigation, in both analysis and simulations. In this paper, we leave this question opened and defer it to our future works.

Our second set of experiments focuses on finding the benefit of our service assignment scheme in comparisons with a reference scheme that does not use Query Aggregation and Service Assignment. We fix the number of services to 50 and vary the number of user applications from 40 to 100. The comparison in average utility is shown in figure 3. While our proposed scheme produces quite stable average utility, that metric in the case of the reference scheme decreases when the number of users increases. Under the low and moderate load conditions (40, 50, 60 users), both schemes perform similarly. But when user demand for services is high (80, 90, 100 users) the proposed scheme outperforms the reference one. There is an interesting fact about the average utility. Since user satisfaction is obtained from utilizing services, average utility reflects both user satisfaction and service utilization. From the experiment result, we can conclude that query aggregation - service assignment scheme improves user satisfaction and service utilization in high load conditions without harming those metrics in low and moderate load conditions.

Comparisons in average utility of successful queries and percentage of failed queries are shown in figure 4 and 5. The average utility of successful queries is the average utility over the number of successful queries. Higher average utility per successful query implies that, on average, users get higher utility from a successful query. In other words, higher average utility per successful query means users are more satisfied with queries' results. Thus, the average utility of successful queries represents user satisfaction in terms of the quality of query result. As can be seen from figure 4 and 5, when user demand is high, query aggregation - service assignment scheme improves both quality of query results and number of successful queries. Under conditions with moderate user demand, improvement of higher quality of query results goes with degradation in the number successful queries.

6. Conclusion

We have presented an approach for exploiting group context information in service discovery for ubiquitous computing systems. Our approach models group context as an optimization problem, called Service Assignment problem. A novel service assignment scheme, called QASA, is introduced for multi-user multi-service ubiquitous computing environments when services are non-shareable. QASA bases on query aggregation and an assignment algorithm that appropriately assign discovered candidate services to requesting users. Our experimental results show that QASA is beneficial to both system and users. With regards to the system performance, QASA can greatly reduce load on semantic service discovery, and can provide more efficient candidate service utilization. From user viewpoints, it improves user satisfaction and quality of query results.

This paper presents initial results of our in progress research works. In near future, we target the shareable service assignment problem and investigate its applications in conflict resolution as well as service discovery. Deeper and more complete investigations on service assignment problem for both shareable and non-shareable cases could also lead to interesting conclusions.

Acknowledgment

This research is supported by the Ubiquitous Computing and Network (UCN) Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

[1] R. Abdur, S. El, and B. James. Semantic-Based context-aware service discovery in pervasive-computing environments. *Proc. of the 1st IEEE Workshop on Service Integration in Pervasive Environments (SIPE) in Conjunction with IEEE Intfl Conf. on Pervasive Services (ICPS). IEEE Press*, 2006.

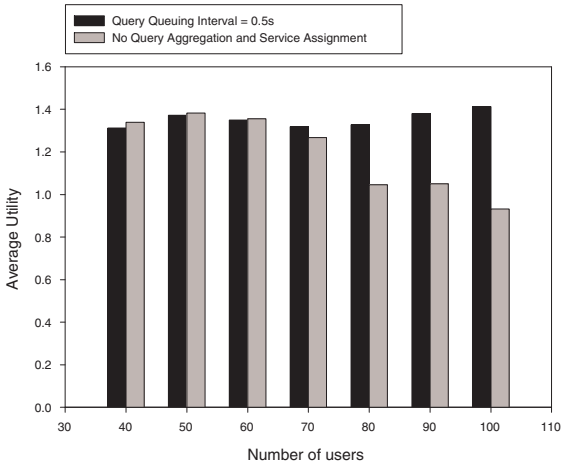

Figure 3. Average Utility vs. number of users

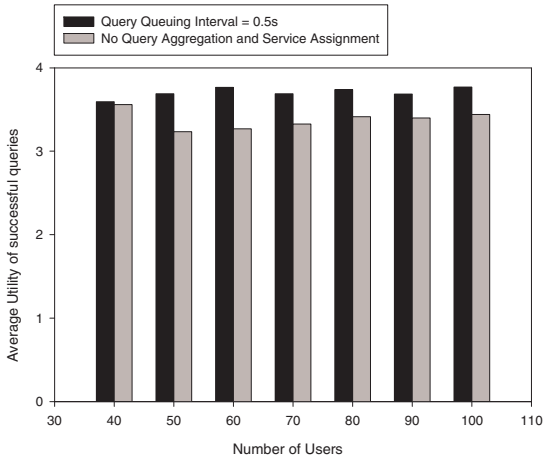

Figure 4. Utility per successful query v.s. number of users

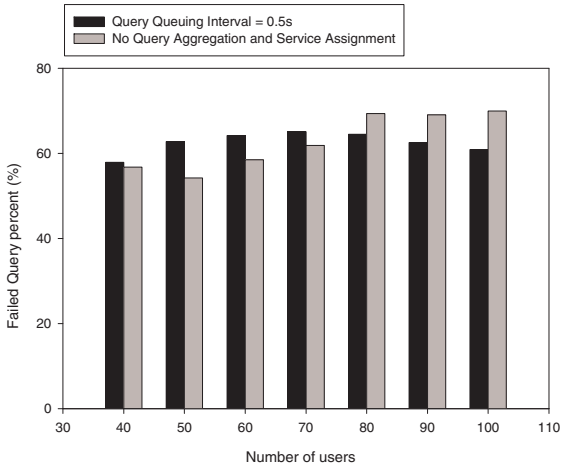

Figure 5. Percentage of failed queries

[2] A. Bernstein and M. Klein. Towards High-Precision Service Retrieval. *First International Semantic Web Conference, ISWC*, pages 84–101, 2002.

[3] D. Bertsekas. The auction algorithm for assignment and other network flow problems. *Computational Optimization and Applications*, 1989.

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

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

[6] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[7] D. Lee. Active surroundings: a group-aware middleware for embedded application systems. *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 404–405, 2004.

[8] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[9] I. Park, D. Lee, and S. J. Hyun. A dynamic context-conflict management scheme for group-aware ubiquitous computing environments. *compsac*, 01:359–364, 2005.

[10] Y. Qi, M. Xi, S. Qi, and J. Zhao. A Conflict Resolution Method in Context-Aware Computing. *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pages 135–140, 2007.

[11] C. Shin and W. Woo. Conflict Resolution Method utilizing Context History for Context-Aware Applications. *Proceedings on ECHISE 2005 held in Conjunction with PERVASIVE 2005*, pages 105–110, 2005.

[12] L. Steller, S. Krishnaswamy, and J. Newmarch. Discovering relevant services in pervasive environments using semantics and context. In *Proceedings of the 3rd International Workshop on Ubiquitous Computing, IWUC 2006, In conjunction with ICEIS 2006, Paphos, Cyprus,*, pages 3–12, 2006.

[13] V. Suraci, S. Mignanti, and A. Aiuto. Context-aware Semantic Service Discovery. *Mobile and Wireless Communications Summit, 2007. 16th IST*, pages 1–5, 2007.

[14] M. Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999.

[15] F. Zhu, M. Mutka, and L. Ni. Service discovery in pervasive computing environments. *Pervasive Computing, IEEE*, 4(4):81–90, 2005.