Towards Service Pool Based Approach for Services Discovery and Subscription

Xuanzhe Liu
Institute of Software
Peking University
Peking, PRC
No. 5, Yiheyuan Road
liuxzh@sei.pku.edu.cn

Li Zhou
Institute of Software
Peking University
Peking, PRC
No. 5, Yiheyuan Road
zhouli04@sei.pku.edu.cn

Gang Huang
Institute of Software
Peking University
Peking, PRC
No. 5, Yiheyuan Road
huanggang@sei.pku.edu.cn

Hong Mei
Institute of Software
Peking University
Peking, PRC
No. 5, Yiheyuan Road
meih@pku.edu.cn

ABSTRACT
In current web service discovery and subscription, consumers must pay too much time on manually selection and cannot easily benefit from the wide QoS spectrum brought by the proliferating services. In our approach, we introduce the service pool as a "virtual service" grouping function identical services together and dispatching consumer requests to the proper service in terms of QoS requirements.

Categories and Subject Descriptors
D.2.12 [**Software Engineering**]: Interoperability-*Distributed Objects*, H.5.2 **[Information Interfaces and Presentation]**: User Interfaces – *Interaction styles, Prototyping*.

General Terms
Management

Keywords
Web services, discovery and subscription, QoS, service pool

1. INTRODUCTION
Nowadays, service discovery and subscription mode not only tires consumers but also prevents them from enjoying high quality of service (QoS) when the services proliferate. On one hand, there may be so many candidate services that have similar or same functionalities but vary from one another on QoS. No matter how the consumers discover their desired services (by keyword searching or category browsing), they have to investigate the details of both services and their providers one by one so that they can select the most proper service (in fact, they select the most proper service provider). On the other hand, a single provider usually serves with a limited QoS spectrum. The consumers have to pay enough attention to find the service provider who can satisfy their QoS requirements. If they cannot find a provider satisfying all their QoS requirements, they usually have to give up all the candidates or make some tradeoff. Furthermore, if the subscribed service becomes unavailable, the consumers have to repeat the above process.

The goal of our work is to simplify consumers' work to discover and subscribe their desired service while satisfying their desired QoS. From our perspective, it is not necessary to list so many candidate providers to the consumers they must browse, select and execute directly. These candidate providers can be clustered as a *"virtual provider"* that alleviates consumers from the time-consuming and tedious selection work: they do not have to view, select and bind to every provider manually, instead, there will be only one service provider serving in a multi-tenancy way. On the other hand, the consumers have the right to enjoy much wider QoS spectrum that a single provider cannot promise. The *virtual provider* clustering several providers may be capable of satisfying the QoS requirements as many as possible.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

2. Usage Scenario
Briefly, there are four phases in QoS aware service discovery and subscription supported by service pool:

1. The consumer logons to our web-based UI and submits the functional requirements to find the services. We support two styles: search by keywords (like Google), and category browsing (like Yahoo! News). Either style will discover several candidate providers. Currently, the category browsing not only requires providers or brokers to build up the categories but also involves consumers in relative complex and tedious human-computer interactions. On the contrary, the keyword search not only puts much less burden on the providers, brokers and consumers but also may find much more candidate providers. In that sense, this paper focuses on the keyword search. As the keyword search may return too many results and some of them may not provide the desired functions, we provide some additional constraints in service similarity matching to improve the precision (discussed in Section 3).
2. The candidate services found are now involved in a service pool. Since the pool can be accessed as a web service, a virtual WSDL will be generated to describe the functions of the pool according to some rules. The mapping rules between the service pool and actual services are generated as well.
3. The QoS values of the services are also recorded, and the pool publishes its QoS spectrum to the consumer. The consumer can submit their QoS requirements within the spectrum. The pool processes the negotiation between different QoS requirements, and discovers the most adequate service.
4. The consumer sends a request to the pool according to the virtual WSDL, the pool routes the request to the actual provider and finally returns the results to the consumer. Note the pool WSDL provided by the service pool is just a description for the functionality that the consumer desires. The pool stores requests from every consumer. Once the consumer finally subscribes the actual provider, the execution engine will look up the discovered provider from step 3 first and route the request to it.

3. Approach Overview

3.1 Service Pool Construction

To represent the different providers as a single service provider to the consumers, we firstly need to cluster the identical providers by retrieving their underlying similarities in topics of both functionalities and domain category:

1. Compute functionality (including operation and input/output) similarity. Here we employ the woogle [1] (which is a web service search engine), to get a set of similar services, denoted as S=<S1, S2,,…, Sn>.
2. Compute domain similarity. Web services search is mainly based on the UDDI registry that is a public broker allowing providers to publish services. We use an important feature in UDDI specification is "***domainkey***" to filter the services into the same domain (e.g., stock services), denoted as SP=<S_1, S_2,..., S_m>, where m≤n.
3. Generate the pool WSDL. Then we sort the services in the pool by computing their cohesion value to the input/output that the consumer uses to search. The cohesion value is computed by using the KMP substring match algorithm.

3.2 QoS-Aware Discovery

In our approach, we choose four QoS properties (price, response time, successability and reputation) specified by the Web Services Quality Modeling [4]. These four important QoS properties may be the most important ones when the consumers use the web services. Considering the real world cases, we furthermore specify two constraints for the QoS values into Hard Constraints (which means the QoS value must be strictly constrained) and Soft Constraints (which means the QoS value can be flexible in an expected range). To make QoS data measured in a standard way, we preprocess the values.

The pool should provide the capability of parsing the consumers' QoS requirements, matching it the most adequate provider and carrying out the execution task. As the selection of the services with multiple QoS attributes has been proved NP-hard [2], we have designed a optimized algorithm to select the most adequate services from the pool in polynomial time. The inputs of the algorithm are the hard and soft constraints for the QoS requirements. The main process of the algorithm is: (1) make the hard constraints of response-time (denoted as *H(RT)*); (2) setup a matrix to record the maximum sucessability value among all selection options with the first k services in the pool and H(RT), and matrix deal with the other three constraints H(RT), H(Price) and S (successability); (3) get the selected services by sorting the service set processed by the matrix in descending order of successability. It can be proved the algorithm complexity is $O(n^2)$.

4. Discussion

The work of service pool is still in progress. We are now working on some important issues.

The first problem is who hosts the service pool. In our work [3], we employ a server that manages the pool in aspects of service clustering, discovery and execution. However, the pool is extremely a "proxy" delegating consumer requests to providers. When the number of providers and consumers grows, the pool would inevitably become a bottleneck and makes the performance degrade. We have taken consideration of alleviating these troubles. To solve the discovery overhead, we identify two different pool types: persistent and personalized. The persistent pool serves for those who have relatively steady requirements for a specific task. The personalized pool is constructed just-in-time for the personalization work and survives only in a session lifecycle. For the persistent pool, the pool server can store the discovered results of steady providers in a cache to alleviate the discovery workload. To reduce the routing overhead, the pool also stores the services stubs in a cache.We use AJAX (Asynchronous Javascript and XML) in the client-side web browser to asynchronously download corresponding stubs just-in-time when the providers are discovered. In this way, the consumers can invoke the service in a end-to-end pattern instead of routing from the pool server.

The basic assumption of the pool is that the services are selected one after another. Considering the constraints for four qualities, the algorithm is still not trivial in an approximate model. This implies that the QoS-aware discovery among thousands of web services may be a very tough challenge. However, our algorithm is specific to limited QoS attributes. To support more QoS constraints at runtime, it still needs some extension. Besides, as a tradeoff for discovering providers in a polynomial time, the sacrificed accuracy may cause problems. These limitations will be considered and improved in our future work.

5. Conclusion and Future Work

Aggregating function-similar web services is considered as a promising way to provide flexible quality of service. The aggregate of web services is usually done by service provider or third party, who cannot precisely and completely predict the requirements and preferences of all service consumers. This paper provides a novel approach that aggregates services from the perspective of consumers instead of providers.

The most important contribution of this paper is to make the aggregation and usage of similar web services on-demand, user-friendly and efficient. On-demand means the aggregate is driven by consumers instead of providers. User-friendly means consumers do not select services, handle different WSDL of similar services and switch between services at runtime any longer. Efficient means it integrates an efficient service search engine, reduces the incorrect services by some filter, discover the aggregate by a polynomial complex algorithm.

6. REFERENCES

[1] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, Jun Zhang. *Similarity Search for Web Services*. Proceedings of the 30th Very Large DataBase (VLDB) conference, Toronto, Canada, 2004.

[2] L.Zeng, B.Benatallah, A.H,et al., QoS-Aware Middleware for Web Services Composition. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol.30, No.5, May 2004.

[3] Gang HUANG, Li ZHOU, Xuanzhe LIU, Hong MEI, Shing-chi Cheung. Performance Aware Service Pool in Dependable Service Oriented Architecture. Journal of Computer Science and Technology, Springer, No.4, 2006.

[4] OASIS: Web Services Quality Model (WSQM) TC. http://www.webkorea.or.kr/pds/data/pds1/WSQM-ver-0.3_20050909143621.doc OASIS, September 2005.