Semantic Web Services Discovery in P2P Environment

Bin Xu, Dewei Chen
Department of Computer Science and Technology, Tsinghua University
xubin@tsinghua.edu.cn, chendw@keg.cs.tsinghua.edu.cn

Abstract

Web services are new paradigm of using the Web. One of emerging challenges is to discover Web services efficiently and precisely. Unfortunately, existing systems like UDDI often have the problems of low availability, low accuracy, and bad performance. In this paper, we propose an approach of semantic Web services discovery in P2P environment. Firstly, Web services are published and deployed in same Web server to guarantee the availability. Secondly, Web servers are organized into groups to form a structured P2P network. Thirdly, for improving the query performance of service discovering, we propose a 2-layers searching algorithm. Finally we developed a prototype system, and tested the system in real P2P test bed of Planet-lab. Experimental results show our approach's precision, recall, query time and scalability. By analysis of the performance, we observed that the system has good efficiency and scalability.

1. Introduction

Web services are emerging as a new paradigm to construct distributed applications in the Web and enable enterprise-wide interoperability. Technologies like WSDL, SOAP and UDDI constitute the current standards of Web services. With the growing popularity of Web services, Web services discovery is becoming a big challenge.

Unfortunately, existing standards of discovering Web services are mainly through UDDI servers, which are focusing on operational and syntactic details to implement and execute Web services. Thus the discovery of Web services is limited to keyword-based search. But facing with thousands of Web services, keyword-based search with manual intervention is a problem. Moreover, after searching in UDDI, not all the Web services listing in the searching result are available. For example, when you search in Microsoft UDDI server, you may get many un-accessible Web services. The problem is due to that UDDI server can not track the real status of each Web services.

In order to discover Web services more efficiently and precisely, it is necessary to improve availability, accuracy and performance. This is exactly the problem addressed in this paper.

In this paper, we propose to conduct Web services discovery in four steps. First, Web services are deployed and published in same Web server. Then we use an algorithm to convert WSDL into OWL-S for each Web Service. We next cluster the Web servers into groups to form a "structured" P2P network. Finally, for improving the query performance of service discovering, we propose a 2-layers searching algorithm.

We have developed a prototype system for P2P Web services discovery based on JXTA (www.jxta.org). We conducted experiment to test the proposed approach. We tried to collect Web services from different sources. In total, 1000 WSDL files are gathered. The experiments ran on Planet-lab (www.planet-lab.org), which is a real P2P test bed in Internet. And we also simulate 2000 peers with plain text description to evaluate the scalability of our approach. The experiments and simulation indicates that our approach can indeed enhance the performance of Web service discovery in P2P environment.

The contributions of this paper are a structure of Web server to tightly integrate the deployment and publication of Web service together to guarantee the availability of Web service, a P2P architecture of service discovery system to share service description in order to improve the recall and performance of service discovery.

The rest of this paper is structured as following: in section 2, we give the related work. In section 3, we discuss the issue in current Web service discovery research. In section 4, we give our approach, presenting the architecture of P2P service discovery system, and introducing the 2-layers searching algorithm. In section 5, we present the experiment and simulation, and analyze the accuracy and performance

2007 International Conference on Parallel Processing Workshops (ICPPW 2007)
0-7695-2934-8/07 $25.00 © 2007 IEEE

of the service discovery system. We make concluding remarks in section 6.

2. Related Work

As we know, current Web service discovery standard of UDDI provides business registry for service provider to publish Web service, and for service requestor to find Web service. Searching for registered Web services is based on keyword matching. Though each UDDI registry can include some nodes to replicate the data, it is basically a centralized design for Web service publishing and searching.

Semantic Web service discovery aims to improve accuracy of discovery by adding semantics to service description. There are two approaches. One approach is to enhance current Web services standards like WSDL and UDDI [1][7][8], to describe capability of Web services by semantics. The other approach utilizes some ontology-based description language like WSMO[3] or OWL-S (formally DAML-S[6]) to describe Web service.

As far as we know, there is some work about P2P Web service discovery. METEOR-S Web Service Discovery Infrastructure (MWSDI) [8] is an infrastructure for Web services publication and discovery. MWSDI integrated P2P technology and UDDI specification to build P2P network between UDDI servers. MWSDI uses a shared Registries Ontology to divide the UDDI servers according to the knowledge domains, and each registry was mapped to the node of Registries Ontology. In literature [2], Thaden etc. use DAML-S service description to provide enhanced semantic search capabilities, and organize all registries into P2P network.

3. Issues of Web Services Discovery

Availability, accuracy and performance are crucial criteria of Web service discovery. Availability is the number of the accessible Web services comparing to the number of all returned Web services in one query. Accuracy is made up of two parts: precision and recall. Precision is the number of correct returned Web services comparing to the number of all returned Web service in one query, while recall is the number of returned correct Web services comparing to the number of all correct Web service. Performance includes the response time for one query and the total network overhead.

Among these criteria, availability is the most important one, because the purpose of service discovery system is to find the accessible Web service. It is useless to get an un-available Web service. And accuracy is also very important, since finding a "correct" Web service is the basic step to access it. Performance concerns about the efficiency of service discovery system.

Low availability is a big issue of current UDDI servers. According to Kim and Rosu's survey[4] of UDDI registry, approximately 67% of the registered Web services are not available or valid. Furthermore, many of the downloaded WSDL files omit mandatory elements or contain other syntax errors. There are two factors affecting the availability of Web service, one is that whether the Web server is running, the other is that whether the WSDL file of this Web service is valid and accessible in the Internet. But in UDDI server, there is no status monitor to insure the Web server is running; even some malicious providers register an un-existing Web service. So, the separation of service provider and service registry might lead to the un-available Web services.

Low accuracy is another issue of current UDDI server, especially the precision is a big issue. Since UDDI is limited to keyword-based searching and lack of powerful search mechanism, precision and recall are low. Adding semantics (like OWL-S) to Web service description in UDDI and WSDL is a good way to improve the accuracy[1][3][6][7][8][9]. But most of the current description of Web service is based on WSDL, how to convert WSDL to OWL-S is a problem.

In P2P Web service discovery, bad performance is also a key issue. Since the query is forwarded in P2P network, sometimes in the manner of flooding, the response time may be too long and network overhead is big. So the topology and query forwarding algorithm may affect the performance of P2P Web service discovery.

4. Our Appoach

4.1. Architecture

To improve the availability of Web service, our approach binds the Web service's deployment and publication in one Web server. Since the publication of Web service is in the same Web server, if the server doesn't work, the service description cannot be accessed; if the server is running and the service description is accessed by service consumer, then the Web service can also be accessed by the consumer. So this insures the availability of Web service. To improve the accuracy, the Web service description file of WSDL is converted to OWL-S file and plain text description automatically, and these descriptions are published in same Web server, not in UDDI server. A

semantic matching algorithm is proposed for match-making of Web service discovery. To improve the performance, Web servers are organized into P2P network according to the description of Web services. Peers with similar service description are formed into one group, and query is forwarded to the groups with similar service description. Then the network overhead and query time may reduce.

In our service discovery system, Web servers are organized into structured P2P network. Web services are deployed on Web server. But there is no central server like UDDI to publish Web services. Each Web server publishes its Web services itself, and discovers other Web server's Web services directly through P2P network.

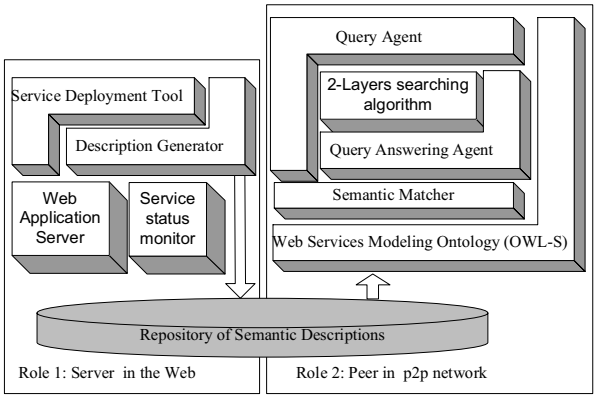

Figure 1. Structure of Web Server.

Figure 1 illustrates the structure of each Web server. There are two roles of Web server. One is server in the Web and the other is peer in P2P network. As a server, service deployment tool is used by the service provider to deploy the Web services on Web application server, then the description generator generates the semantic descriptions of OWL-S. The service status monitor keeps deployment and publication status consistent. Web services on Web application server are accessed through current standards (SOAP and HTTP).

As a peer, repository keeps the semantic descriptions. All peers share the same Web services modeling ontology and use the same semantic matcher. Query agent collects queries from users and forwards them in P2P network. It also receives the query results and displays them to user. Query answering agent responds to the query through semantic matching, and continues to forward the query. Query agent and query answering agent utilize 2-layers searching algorithm.

4.2. Generate Description

Compared to UDDI, our discovery system describes the Web services not only in WSDL, but also in OWL-S and plain text. The component of description generator creates OWL-S file for each Web service. Usually Web service has a WSDL file. The generator converts the WSDL file into OWL-S files via an annotation algorithm. The annotation algorithm of our work is discussed in literature [5].

Our discovery system not only generates semantic description of Web services, but also generates plain text description of Web service. In WSDL file, there is a tag <documentation> describing the function of each operation. So, plain text description is generated by parsing the tag <documentation>. There are two advantages of plain text description: one is to help the user in understanding the Web services, and the other is to form a structured P2P network.

4.3. Structure P2P Network

In P2P network, to reduce the query time and raise the query precision, the query should be forwarded to related peers. To structure the P2P network, we use groups to divide the P2P network into two layers. Layer one is made up of many peer groups, while layer two is made up of many peers inside one peer group.

Each Web server might have more than one Web service, and each Web service has its plain text description. So, we extract keywords from all plain text description of one server, and forms Vector *VP* to describe the server (peer): (w_i is the term weight [11] of keyword k_i)

$VP=\{<k_1,w_1>,<k_2,w_2>, \ldots<k_i,w_i>\}$

Thus the similarity between Peer1 and Peer2 is:

$Sim(Peer1,Peer2)=Similarity(VP1,VP2)$

Several peers form a group through the clustering of VP. One peer of the group acts as group server to keep the description of the group, which is defined by vector VG:

$VG=\{<k_1,w_1>,<k_2,w_2>, \ldots<k_g,w_g>\}$

The statistics of k_g and w_g are according to every peer's *VP* in the group.

Accordingly, the similarity between Group1 and Group2 is:

$Sim(Group1,Group2)=Similarity(VG1,VG2)$

And the similarity between Group1 and Peer1 is:

$Sim(Group1,Peer1)=Similarity(VG1,VP1)$

The similarity between query R and Group is:

$Sim(R,Group)=Similarity(K_R,VG)$

As figure 2 illustrated, the topology of P2P network is made up of peer groups. One peer joins one group if the *Sim(Group,Peer)* is higher than a given threshold. Communication inside group is through broadcasting, while communication outside group is through peer group servers. Each peer keeps its peer vector *VP*, while peer group server keeps not only its peer vector

VP but also the group vector *VG* and route table. Route table represents 'acquaintances' of the group. It keeps the most similar groups and least similar groups, according to the ranking of similarity between the peer's group vector *VG* and other groups' *VG*.

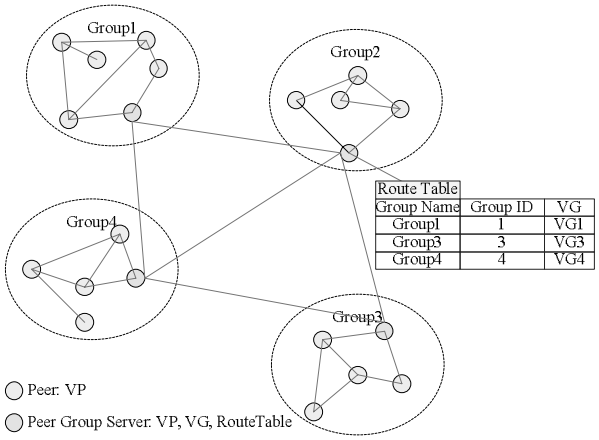

Figure 2. Topology of P2P network.

4.4. Query Process

Every query is made up of two files. One is owl file for semantic matching, and the other is plain text file for routing. For instance, to discover Web services about weather forecast, the query is:

Query.txt: "weather query check weather by specified date and city weather query services".

Query.owl:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
 <profile
 xmlns="http://keg.cs.tsinghua.edu.cn/ontology/travel#"
 ……
 >
 <profile:serviceName>weather query
 </profile:serviceName>
 <profile:textDescription>check weather by specified
date and city.</profile:textDescription>
 ……
 <profile:hasInput rdf:resource="#Date"/>
 <profile:hasInput rdf:resource="#City"/>
 <profile:hasOutput rdf:resource="#Weather"/>
 <profile:hasPrecondition></profile:hasPrecondition>
 <profile:hasResult></profile:hasResult>
 </profile>
```

In query.owl, the input and output are classes of travel ontology. The query.owl file is used in semantic matching, while the query.txt file is used to route the query.owl file to according peer groups. We extract keywords from query.txt file, and form query vector:

$K_R=\{<k_1^R,1>,<k_2^R,1>, \ldots<k_m^R,1>\}$

In $<k_m^R,1>$, k_m^R is a keyword, and 1 is the weight of that keyword.

The query is forwarded in two layers: layer one is between group servers, layer two is inside one group.

4.4.1. Layer One Searching. Layer one searching is to locate peer groups which probably satisfy the query. When the query agent of one peer gets the query R, it forwards the query statement like *<Peer-ReqID, R>* to its peer group server. Then the server decides which groups to forward the query, as following:

1) Compute the similarities between K_R and every *VG* in route table, and get the result $SIM=\{Sim_1,Sim_2,\ldots,Sim_i\}$.

2) For every Sim_j in SIM, if Sim_j is higher than $Sim_{threshold}$, then the statement *<<GroupID>*, hops, *<Peer-ReqID, R>>* is forwarded to $Group_j$.

3) If there is no Sim_j higher than $Sim_{threshold}$, then choose most similar groups of quantity Q to forward the statement.

4) While forwarding the statement, current group ID is add to the groups list *<GroupID>*, and hops=hops+1. If hops reach the MaxHops, the statement is not forwarded.

5) If query statement is from other group, and $Group_j$ is in the groups list *<GroupID>*, then the statement *<<GroupID>*, hops, *<Peer-ReqID, R>>* is not forwarded to $Group_j$.

This process utilizes keywords to filter the peer groups, in order to reduce the forwarding peers. Setp 4 and 5 prevent the forwarding from cycle and un-limitation.

4.4.2. Layer Two Searching. Layer two searching is to locate peers which probably satisfy the query. When peer group server receives the query forwarded from other group, it extracts the query statement *<Peer-ReqID, R>*, and broadcasts it to all peers in the group. Each peer in the group receives the query statement and uses its query answering agent to match based on semantic matching. Matched Web services are returned to Peer-ReqID directly in the statement of *<Peer-AnsID, URL, Matched>*. Our semantic matching algorithm is discussed in literature [10].

5. Experiment and Simulation

We implement the service discovery system with Java and JXTA, and run the system in the P2P test bed of Planet-lab. The P2P architecture is implemented with JXTA, and the semantic matcher is implemented with Jena (jena.sourceforge.net).

5.1. Precision and Query Time

We tried to collect Web services for experiments from as many sources as possible. We randomly chose 1000 WSDL files from the www.xmethods.com, www.google.com and www.baidu.com. Among them there are only 74 files are about travel. We annotated these 74 files with travel ontology and generated 74 OWL-S files.

To test the precision and query time in the environment of real Internet, we ran the system in 50 peers of Planet-lab. There are totally 74 Web services about travel. Each Web service has its OWL-S file and plain text file. These Web services descriptions are distributed in 50 peers, and each peer has about six Web services descriptions. We made a query like section 4.4 for more than ten times, and recorded the average precision and query time. To compare with the query time and precision of UDDI, we searched Web services by using the keyword "weather" in Microsoft UDDI server for more than ten times, and also recorded the average precision and query time.

Table 1. Precision and Query Time

	P2P Service Discovery	UDDI
Precision	100%	60%
Query Time (ms)	52558	3213

As the above table illustrated, semantic matching improves the precision to 100%, but cost much query time(averaged 52588 milliseconds). Query in UDDI server is quick, but in the 35 list result of weather Web services, only 21 is available. So, semantic matching improves the precision of Web services discovery, but cost much time. The query time of P2P semantic discovery system is longer than that of UDDI server, because UDDI is centralized design to reduce query time, and keyword-based query of UDDI costs less time than semantic matching. Anyway, the precision of our approach is much better than that of UDDI.

5.2. Recall and Query Time

It is the P2P topology that affects the recall of query. If the query is forwarded to every peer group, the recall is 100%, but this will cause network congestion and cost much time. To reduce network overhead and query time, the query should be forwarded to the groups with similar description. It is very important to make peers with similar description form in one group.

The P2P topology is decided by the clustering of peer keyword vectors, in other words, it is decided by the plain text description of Web services. We can get plain text description from the total 1000 WSDL files, but since they belong to many different domains, the clustering of these files is not good. At the end, to simulate the plain text description of Web services and make them clustering, we generate 2000 text files; each text file describes one Web service. Every 40 text files are put in one peer, then there are totally 50 peers. In order to make the peers clustering, the 2000 text files were generated like this:

1) We got a word set of 5075 words from dictionary.

2) The 5075 words were divided into 10 sets. Each set had about 600 words.

3) Each set generated 200 text files, and each text file had 50 words randomly selected from the 600 words.

Every 40 text files in one set were put in one peer, so there are five peers having the text files from one set. This made these five peers have similar *VP* (peer vector), and form into one peer group finally. There were totally 10 peer groups.

We put all these 2000 text files for 50 peers in Planet-lab to test the query time and recall. The query text is that: "The Web services are about travel, which help me to book flight ticket, reserve room in hotel, reserve car, or even forecast the weather."

The same query was committed 5 times, and got the result as following.

Table 2. Recall and Query Time

Query1	Recall	0.14	0.28	0.44	0.61	0.75	1.00
	Time	312	609	656	1219	1984	2015
Query2	Recall	0.14	0.31	0.44	0.61	0.75	1.00
	Time	360	485	531	594	953	1922
Query3	Recall	0.14	0.31	0.44	0.50	0.67	1.00
	Time	391	750	1079	1891	1922	2000
Query4	Recall	0.14	0.31	0.44	0.58	0.75	1.00
	Time	344	563	594	781	1375	1594
Query5	Recall	0.14	0.31	0.44	0.61	0.75	1.00
	Time	328	546	640	953	1125	1187

According to above table, every query reaches the recall of 75% in no less than 1984ms. At the time of 1000ms, the averaged recall is about 56%. This means that 56% matched Web services return in one second. Since each peer should set a time interval to get the returned service, the setting of 2000ms should guarantee 100% matched services to return. Because Planet-lab is a real Internet environment, our P2P semantic discovery system proves its efficiency in the Internet.

5.3. Recall and Scalability

With the quantity of peer increases, scalability of the discovery system may affect the recall. To simulate the discovery system in large scale peers, we generated plain text description of Web service like section5.2 for 100 peers, 1000 peers and 2000 peers respectively. Section4.4.1 details the layer one searching which will decide the quantity of the peer groups to forward the query. Forwarding query to less peer groups may improve the performance of service discovery, but it may lead to low recall. Search scale is the quantity of peer groups chosen to forward the query compared to all peer groups. According to different search scale of layer one searching, we committed five queries and get the average recall for different scale of 100 peers, 1000 peers and 2000 peers accordingly.

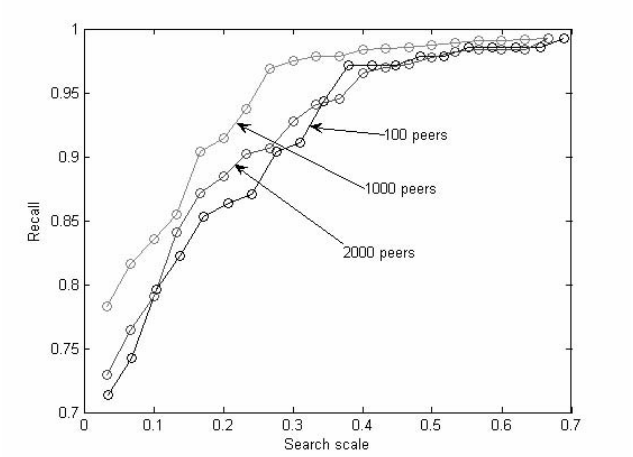

Figure 3. Recall and Search Scale

From above figure we can see that the three curves are very near, that means the recall increases similarly while the peer number increases. When the search scale is only 20%, the recalls of the three scales are more than 85%. In other words, with the network overhead in 20% peer groups, the discovery system can get 85% recall. The increasing of peer number doesn't affect the recall so much, because the P2P network is structured with groups; and new peer may add to the most similar peer group. So, the discovery system is scalable while keeping a good recall.

6. Conclusion

In this paper, we have investigated the issue of Web service discovery. We have proposed an approach for discovering semantic Web services in P2P network. Web services are deployed and published in same Web server, and description of Web services in OWL-S file and plain text are generated and stored in P2P network. We present a 2-layers searching algorithm to control the message forwarding complexity in P2P network. Experimental and simulative results show our approach's precision, recall, query time and scalability. By analysis of the performance, we observed that the system has good efficiency and scalability.

7. References

[1] K. Sivashanmugam, K. Verma, A. Sheth, J. Miller. Adding Semantics to Web Services Standards. Proceeding of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada ,June 2003, pp.395 - 401.

[2] U. Thaden, W. Siberski, W. Nejdl, A Semantic Web based Peer-to-Peer Service Registry Network, Technical Report, Learning Lab Lower Saxony, 2003.

[3] D. Roman, H. Lausen, U. Keller, et al. D2v1.1. Web Servcie Modeling Ontology (WSMO), WSMO Working Draft 10 October 2004. http://www.wsmo.org/2004/d2/v1.1/20041010/.

[4] S.M. Kim, M. Rosu.b A survey of public Web servcies. WWW2004, NY, USA, May 2004, pp. 312-313.

[5] D. Zhang, J. Li, B. Xu. Web Service Annotation Using Ontology Mapping. Proceeding of IEEE International Workshop on Service-Oriented System Engineering, Beijing, China, Oct 2005.

[6] A. Ankolekar, M. Burstein, J. Hobbs, et al. DAML-S: Semantic markup for Web services. Proceeding of the First International Semantic Web Working Symposium (SWWS): Infrastructure and Applications for the Semantic Web, Stanford University, CA, USA, 2001, pp. 411--430.

[7] N. Srinivasan, M. Paolucci, K. Sycara. Adding OWL-S to UDDI, implementation and throughput, SWSWPC 2004 6-9, 2004, San Diego, California, USA.

[8] K. Verma, K. Sivashanmugam, A. Sheth, et al. METEOR–S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management, 2004.

[9] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. Proceeding of the 1st International Semantic Web Conference (ISWC), Sardinia, Italia, June 2002, pp. 333-348.

[10] P. Zhang, J. Li, K. Wang. Discovery and Query: Two Semantic Processes for Web Services. The 5th IFIP conference on e-Commerce, e-Business, and e-Government (I3E 2005), Poznan, Poland, Oct 2005.

[11] G. Salton, C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing& Management, 24(5), pp. 513-523, 1988