Scalable Web Service Discovery on P2P Overlay Network

Gang Zhou\textsuperscript{1,2} Jianjun Yu\textsuperscript{1} Rui Chen\textsuperscript{1} Hui Zhang\textsuperscript{1}
\textsuperscript{1}State Key Lab. of Software Development Environment, Beihang University, 100083
\textsuperscript{2}National Digital Switching System Engineering & Technological Research Center, 450002
\{gzhou,yujj,chenrui,hzhang\}@nlsde.buaa.edu.cn

Abstract

Decentralized approaches for Web Services discovery, such as Peer-to-Peer, become more and more attention-getting in the scientific community. In this paper, we propose a peer-to-peer framework, which adopts an enhanced Skip Graph named ServiceIndex as the overlay network for service discovery. To guarantee discovery efficiency, ServiceIndex schemes WSDL-S (Web Services Semantics) as Semantic Web Services description language and extracted its semantic attributes as indexing keys in Skip Graph. Also a multi-layer P2P overlay network was constructed to aggregate similar keys and keep load balancing on peer nodes. The evaluation showed that the ServiceIndex system performs considerable service discovery efficiency.

1. Introduction

The Web Services (abbreviated as service) are designed to support the reuse and interoperation of software components on the web, and are receiving ever increasing interests from consumer, e-commerce, and research communities across different areas. A fundamental problem of Web Services concerns service discovery.

Service discovery is to match service requirement and service capability. Service requirement is originated from service consumers who want to complete Internet-based tasks. They hope to use complex but flexible search mechanism to get exact and needed services. Service capability depicts the functionality (e.g., input, output, precondition, and effect) and non-functionality (e.g., Quality of Service) of the entity. It is always described by xml-based language, such as WSDL (Web Service Description Language) protocol. In this paper, we categorize current service discovery systems into two types: service inquiry and service matching. Service inquiry uses WSDL/ UDDI (Universal Description, Discovery, and Integration) protocols to find syntactic descriptions and gives extensive but imprecise results. Service matching uses OWL-S (OWL Web Ontology Language for Services), WSMO (Web Service Modeling Ontology) and etc. to present service semantics. Furthermore, it provides matching engine to support accurate and provable inference of search results. These two types rely either on the centralized way or the distributed way, which leads research on service discovery into four directions: syntactic centralized discovery, syntactic distributed discovery, semantic centralized discovery and semantic distributed discovery.

Most current systems focus on the first three directions, which remain difficulty in supporting complex search (e.g., range query and semantic matching) of service descriptions while keeping system’s scalability, fault-tolerance, stabilization, and accuracy. P2P based semantic service discovery system with its supporting scalability, range query, and semantic search is prevailing nowadays. In this paper, we proposed a novel P2P semantic discovery system named ServiceIndex. It integrated the first three mechanisms to provide extensive service inquiry and accurate service matching. The main contribution of ServiceIndex system is as follows. First, ServiceIndex committed to the use of WSDL-S (Web Services Semantics) for describing Web Services because of its enriched semantic capabilities. The semantic elements or attributes in WSDL-S documents are then extracted as keywords to index the services in P2P overlay network. Second, ServiceIndex constructed an enhanced multi-layer Skip Graph P2P overlay network. This overlay has three advantages: It distributes the service description in dictionary order and thus supports prefix search; It preserves locality of WSDL-S descriptions and thus supports range query on these partial WSDL-S segments; It aggregates the similar keys and thus supports better range query and load balancing. This new system provides a novel architecture to classify massive loose-coupled and heterogeneous services on Web. It enables easier service publishing for service provider, and more quickly, succinctly service discovery, integration and invocation for service consumer.

The rest of the paper is organized as follows. The second part presents the related work. The third part introduces the partition algorithm of WSDL-S segments. The fourth part illustrates the Skip Graph P2P overlay network and its en-
hancement suitable for service inquiry. The fifth part shows the service discovery on semantic service descriptions. The sixth part gives the experiment results. The final part is conclusion and future work.

2. Related Work

As illustrated above, there are four directions of service discovery. In this section, we present the current systems and work as follows: UDDI based service discovery, semantic service discovery and P2P based service discovery.

2.1 UDDI Discovery

As an indispensable role in Service-Oriented Architecture, UDDI provides a universal service description, discovery and integration mechanism that tries to ease providers publishing and consumers discovering services. Whereas this centralized registry meets insurmountable problems such as single point of failures, delayed delivery of notifying updated service description, and lack of semantic association when applied to Internet for large scale of service discovery. Figure 1 shows services’ status registered in current Microsoft UDDI registry provided at 2007-01-25.

![Figure 1. Active Services vs. Total Services](image)

UDDI v3 supports federated repositories that provide a distributed service discovery environment, whereas it publishes updated service descriptions to all federated repositories. This operation certainly would cause redundancy and network congestion.

2.2 Semantic Discovery

Semantic Web Services are expected to allow automatic service discovery, invocation, composition, monitoring, and interoperability. Current systems [7, 8, 10] on semantic service discovery are always based on centralized mechanism. This mechanism is suitable for small scale of service discovery whereas causes inevitable problems when applied to Internet-scale service discovery due to its low efficiency of matching engine.

2.3 P2P Discovery

Web Services description language in nature is an xml-based language. Discovering Web Services in a P2P overlay network is the process of searching distributed and structural xml information comparing to traditional distributed resource discovery provided by previous P2P systems. In this paper, we category this distributed and structural xml information search to complex search such as range query, prefix query and wildcard key query.

pSearch [13] distributes documents through the CAN overlay based on document semantics generated by Latent Semantic Indexing (LSI) which computes cosine of the angle between vectors $v_a$ and $v_q$ to get the similarity. Upon reaching the destination, the query is flooded to nodes within a radius $r$, determined by the similarity threshold or the number of wanted documents. The report, in [11], describes a dimension reducing indexing scheme that effectively maps the multi-dimensional space to physical peers with a locality-preserving mapping called Space Filling Curves (SFC). In [4], to support range query and load balancing, RST(Range Search Tree) and LBM(Load Balancing Matrix) are provided for content registration and query resolution in DHT-based system. Also, work in [2, 6, 14] provides a feasible framework for service discovery. However, these systems are mostly based on DHT systems and only support searching in hashed key space which is flat and not applicable for complex search on keywords. Skip Graph [1, 9] based systems can support range query [3, 12], load balancing, high fault-tolerance [5], locality-preserving and tree lookup [15] for web resources. We studied these approaches and proposed the ServiceIndex system for service distribution and discovery.

3. WSDL Partition

Our P2P indexing system is similar to other P2P systems which use keywords as indexing keys to complete resources lookup. That means metadata in service descriptions should be extracted as keywords to index services in our system. To support complex search and semantic query, WSDL-S is schemed as service description language. We first study WSDL-S and analyze its advantages for service discovery, then introduce the methodology of keywords’ partition from WSDL-S documents.

WSDL-S \(^1\) adds extensible elements and attributes in WSDL to denote semantic information of the service. Elements are added with “wssem:modelReference” attribute to denote semantic information imported from OWL (Web Ontology Language). In addition, it can be recognized by WSDL parsers just as traditional WSDL documents and in-

\(^1\)http://www.w3.org/Submission/WSDL-S/
 interoperable with other services. Figure 2 gives a WSDL-S fragment from the service OrderService expressed by WSDL-S.

```xml
<operation name="processPurchaseOrder"
  wssem:modelReference="Rosetta:RequestPurchaseOrder">
  <input messageLabel="processPurchaseOrderRequest"
    element="tns:processPurchaseOrderRequest"/>
  <output messageLabel="processPurchaseOrderResponse"
    element="processPurchaseOrderResponse"/>
  <wssem:precondition name="ExistingAccountPrecond"
    wssem:modelReference="POOntology#AccountExists"/>
  <wssem:effect name="ItemReservedEffect"
    wssem:modelReference="POOntology#ItemReserved"/>
</operation>
```

Figure 2. A WSDL-S Fragment of OrderService

WSDL-S adds semantic, accurate characteristics to syntactic Web Services technology which helps to match the service precisely via ontology and inference mechanism. Since the functionality of a service is expressed in the "interface" element including "input," "output," "precondition," and "effect" on which matching engine relays for service accurate matching. In this paper, we only extract these elements and sub-elements into P2P overlay network to index services.

3.1 Partition Algorithm

The elements and attributes of WSDL-S are typically organized in hierarchical trees to enable similarity indexing and queries in centralized search engine. To preserve locality, tree lookups, and search the partitioned tree efficiently in P2P overlay network, we linearize the tree structure of WSDL-S by pre-order traversal. Table 1 illustrates how to split the above service "OrderService" into six segments.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>inf</td>
<td>interface@PurchaseOrder</td>
</tr>
<tr>
<td>opr</td>
<td>inf@operation@RequestPurchaseOrder</td>
</tr>
<tr>
<td>int</td>
<td>opr@input@processPurchaseOrderRequest</td>
</tr>
<tr>
<td>out</td>
<td>opr@output@processPurchaseOrderResponse</td>
</tr>
<tr>
<td>pre</td>
<td>opr@precondition@POOntology#AccountExists</td>
</tr>
<tr>
<td>eff</td>
<td>opr@effect@POOntology#ItemReserved</td>
</tr>
</tbody>
</table>

WSDL-S schema is not changed, which makes the elements and attributes well acknowledged to all service providers and service consumers. So we give the abbreviations to these elements and attributes as shown in Table 1. For example, we can define "inf@PurchaseOrder/opr@RequestPurchaseOrder" to represent the segment split from "operation" element.

The linearization had two advantages. First, it does not destroy the structure of service descriptions. We can restore these segments to the original WSDL-S. Second, it can preserve the locality of elements. That is, sub-elements could use segment of their parent element as prefix.

For any WSDL-S document's partition, we use the following algorithm to generate segments:

1) Extracts the semantic value if exists, else extracts the first attribute of current element.
2) Uses "@" to separate the attribute value from its owner element, "/" to compose the element with its all parent elements.
3) If sub-elements or sub-attributes of current element can not be split, uses the segment of current element to identify them.

Applying the above algorithm, we will extract at least six segments from a simple service which has only one "interface" element. Whereas it would generate too many segments from a complicated service with many "interface." To keep the load balancing of the system, we apply 3) to reduce the number of segments according the characteristic of locality preserving. By this means, a segment would conclude a list of tree nodes of WSDL-S document tree. For example, if we do not split the "input" element, it will be represented by the segment split from its parent element "operation." The partition algorithm preserves locality since the sub-elements or sub-attributes are combined with their parent elements. For example, if service consumers make an inquiry on element "input" in P2P overlay network, he would find the desired keywords are distributed close with the key "operation" which is the parent of "input." For simplicity, we adopt these segments as the keys of peer nodes in Skip Graph, and make the service URL as values.

4. P2P Overlay

In the above section, we obtain (key,value) pairs from WSDL-S documents. The next step is to distribute these keys in the peers of an overlay network. This step is similar to the existing current P2P systems. In ServiceIndex system, we use an enhanced Skip Graph to construct the P2P overlay network described as follows.

4.1 Introduction to Skip Graph

Skip Graph presents a distributed data structure for searching ordered data in a peer-to-peer network. It is a randomized, balanced-tree data structure by adding redundant connectivity and multiple handles for locality-preserving distribution of tree structures which provides a probabilistic guarantee that the standard ordered dictionary operations
can be performed in $O(\log N)$ time within $n$ nodes system. All keys appear in sorted order in the double-linked list at Level 0, but each Level $i$ ($i > 0$), can contain multiple double-linked lists. Each peer node maintains a membership vector $M(x)$ which is a random string of bits over fixed alphabet. A list at Level $i$ contains all keys that have the same prefix of length $i$ for their $M(x)$ which continues until the key becomes a singleton, so it will result in, on average, $O(\log N)$ levels in the Skip Graph. When searching, the query is started at the top level and continued at lower level if required until to level 0. The address of the peer node storing the key or closest to the search key is returned. Figure 3 gives a simple Skip Graph with 3 levels.

![Figure 3. A Simple Skip Graph](image)

If we originate a query to find the destination key "42" at peer node "19," the query message is first routed to the neighbor node "30" at level 2. At node "30," the message is routed to the node with the destination key "42" at level 0.

Like DHT systems, Skip Graph scales gracefully, and offers excellent query complexity. Skip Graph can be applied to locality-sensitive applications (e.g., distributed file system service), xml-based document retrieval (e.g., service discovery), and multi-dimensional indexing (e.g., geographic information systems and multimedia databases) in distributed environment.

We adopt Skip Graph as P2P overlay network for several reasons. First, Skip Graph has an advantage over DHT in the sense that it directly supports range query, while DHT only provides exact search. Second it supports locality preserving for similar keys to keep semantics of services. The third is that it can distribute a WSDL-S document to variable peers with different granularity of partition.

### 4.2 Constructing P2P Overlay

P2P overlay is dynamic and unstable. Constructing a P2P overlay network means to concern two distinct types of operations: key maintenance and node maintenance. Key maintenance includes search, insertion, update, and deletion. Node maintenance includes node joins and departures. These operations are described below.

**Key search:** This operation enhances the search operation in Skip Graph at two aspects. First, the same segments from different WSDL-S documents are allowed. Second, we would return the nearest value with the given key if no exact key is found. It will take $O(\log N)$ time and $O(\log N)$ ($n$ is the number of peer nodes in Skip Graph) messages to complete key search without considering concurrency.

**Key insertion and deletion:** The insertion operation is almost the same as the insertOp in Skip Graph except that the link operation should take into account the duplication of keys. Our modified link operation just inserts the value with the same key into the existing node. The deletion operation follows the same steps to delete the useless keys. It will take $O(\log N)$ time and $O(\log N)$ messages for description insertion and deletion.

**Key update:** When updating service description, we should apply search operation to find the tree node with this given key, and then use the update operation to replace the old value with the new one.

**Node join and departure:** Two circumstances will bring on the node join operation. The first is that a new key can not be inserted into any current peer nodes. It will take $O(\log N)$ time and $O(\log N)$ messages to complete this operation. The second is that the current nodes in the Skip Graph are being hot spots. We create new nodes to keep load balancing of Skip Graph. It will take approximately $O(\log N)$ time and $O(\log N)$ messages to complete this operation. When a node leaves, we should apply repair mechanism to restore the connectivity of Skip Graph and check missing segments of the WSDL-S document. Then retrieve the failure one and re-distribute it into Skip Graph. Similar time and messages should be consumed for node departure operation.

### 4.3 Enhancements

In ServiceIndex, we make the similar keys appear at the same node and construct a multi-layer Skip Graph shown in Figure 4.

![Figure 4. The Structure of Multilayer Skip Graph](image)

We adopt a simple principle to insert keys into the multi-layer Skip Graph to achieve better load balancing and more
efficient range query: keys with the longest common prefix will be inserted into the same Skip List (named as ServiceBag) and the smallest of these keys is elected as representative key of peer node in Skip Graph. A ServiceBag includes all the keys in the scope of itself. To determine the scope of a ServiceBag, we need to define the following two parameters:

- RK: For a ServiceBag \( S \), RK is its representative key which equals to the centroid of all the keys in \( S \). In Skip Graph layer, only the RK of \( S \) appears as the key of peer node.

- Radius: A ServiceBag needs a Radius \( r \) to determine the boundaries of its scope. It means that, for a ServiceBag \( S \) with RK \( r \) and \( r \), keys within \([RK−r, RK+r]\) will be in \( S \).

As described above, when a key is to be inserted into the system, first we need to find the right ServiceBag the key belongs to, and then put it in. The new inserted key will lead to some influence on the target ServiceBag. At first, the RK of the ServiceBag will be changed because of its recalculation, according to the definition described above. And at the same time, the scope of the ServiceBag also moves accordingly, which is called excursion phenomenon illustrated at Figure 5. The excursion of the scope can produce domino effect. For example, the movement of a ServiceBag may lead to the overlapping of the scopes of its neighbor ServiceBag and itself, which is rebellious for the searching process in a distributed system. The ServiceBag excursion takes place when a key is inserted, changed and deleted. To address the issue, we propose an in-depth model, which is called excursion phenomenon illustrated at Figure 5. The excursion of the scope can produce domino effect. For example, the movement of a ServiceBag may lead to the overlapping of the scopes of its neighbor ServiceBag and itself, which is rebellious for the searching process in a distributed system. The ServiceBag excursion takes place when a key is inserted, changed and deleted.

To determine the right way in which ServiceBag moves. The details of the model is described as follows.

![Figure 5. Centroid Excursion](image)

**Definition 1:** For binary relationship \(<\)

\[ RK_i < RK_j, \text{iff } \forall x \forall y (x \in S_i \land y \in S_j \rightarrow x < y) \]

where \( S_i \) and \( S_j \) are two ServiceBags, \( RK_i \) and \( RK_j \) are corresponding RKs of the two, \( x \) and \( y \) are the numerical value of the keys.

**Definition 2:** The universal numerical value of the key is expressed as

\[
Uni(key) = \sum_{i=0}^{\|key\|−1} d^{Max\text{length}−i−1} \cdot \text{Num}(key[i]) \tag{1}
\]

where \( \|key\| \) is the length of the key, \( \text{Num}(key[i]) \) is the numerical value of the \( i \)th character of the key. \( Max\text{length} \) is the maximum length of key, \( 1 \leq \|key\| \leq Max\text{length} \). So a key can be regarded as a universal \( d \)-radix integer and the similarity between two keys can be expressed as \( Sim(key_1, key_2) \), which is the absolute difference between the numerical value of two keys. In this model, we normalized the value of \( Sim \) in \([0,1]\).

**Definition 3:** Assume that, all keys in the system form the data set \( S = \{key_1, key_2, \ldots\} \), and there are \( n \) ServiceBags denoted as \( S_1, S_2, \ldots, S_n \) with corresponding radius \( \sigma_1, \sigma_2, \ldots, \sigma_n \). The centroid of \( S_i \) is:

\[
m_i = \frac{1}{\|S_i\|} \sum_{x \in S_i} x
\]

when a new key \( key_{new} \) is inserted into the system, we should firstly find the neighbors \( R_k \) and \( R_{k+1} \), which satisfies \( RK_k < key_{new} < RK_{k+1} \). Then we calculate \( Sim(key_{new}, RK_k) \) and \( Sim(key_{new}, RK_{k+1}) \). The following is the algorithm to insert this new key:

1) If \( Sim(key_{new}, RK_k) \leq r_k, Sim(key_{new}, RK_{k+1}) > r_{k+1} \), insert this new key into ServiceBag \( S_k \). Then apply following changes (Figure 6(a)):

- \( S_k \leftarrow S_k \cup \{key_{new}\} \),
- \( m_k \leftarrow \frac{(\|S_k\|−1)m_k+x_{new}}{\|S_k\|} \),
- \( r_{temp} \leftarrow \text{Max}\{r_k, \text{Max}\{Sim(m_k, x_j)\}\} \)

where \( x_j \in S_j \).

a. If \( m_k > RK_k \), there are two cases.

If \( m_k + r_{temp} < RK_{k+1} - r_{k+1} \), which means that the updated ServiceBag \( S_k \) will not intersect with \( S_{k+1} \) (Figure 6(a1)), let

\[
RK_k \leftarrow m_k, \ r_k \leftarrow r_{temp}
\]

Otherwise, to avoid intersecting with \( S_{k+1} \) (Figure 6(a2)), let

\[
r_k \leftarrow \text{Max}\{\frac{RK_{k+1} - r_{k+1} - \text{Min}(x)}{2}, r_k\},
\]

\[
RK_k \leftarrow RK_{k+1} - r_{k+1} - r_k.
\]
b. If \( m_k = RK_k \) (Figure 6(b)), keep \( RK_k \) and \( r_k \) unchanged.

c. If \( m_k < RK_k \), there are also two cases:
If \( m_k - r_{temp} > RK_{k-1} - r_{k-1} \), which means the updated ServiceBag \( S_k \) will not intersect with \( S_{k-1} \) (Figure 6(c1)), let
\[
       RK_k \leftarrow m_k, r_k \leftarrow r_{temp}.
\]

Otherwise, to avoid intersecting with \( S_{k-1} \) (Figure 6(c2)), let
\[
r_k \leftarrow \max\left\{ \frac{\max\{x_i\} - (RK_{k-1} + r_{k-1})}{2} , r_k \right\}, \]
\[
       RK_k \leftarrow RK_{k-1} + r_{k-1} + r_k.
\]

\[
       S_{new} = \{ x_{new} \},
\]
\[
       RK_{new} = x_{new},
\]
\[
r_{new} = \min\{ r, RK_{k+1} - r_{k+1} - x_{new}, x_{new} - RK_k - r_k \}.
\]

Figure 7 presents the four cases when a new ServiceBag is created with different radius \( r \).

According to the above process, the radius of a ServiceBag will expand fast, which means there will be too many keys inserted into one ServiceBag to achieve good load balancing. To solve this problem, we propose the half-and-half policy for ServiceBag-split, that is:

3) If the size of a ServiceBag increases beyond the threshold \( t \), the existing ServiceBag will be split into two ServiceBags with the same number of keys.

5. Service Discovery

There are two major steps in the ServiceIndex system for services discovery: finds extensive services in Skip Graph and matches exact services in matching engine.

Service inquiry is to retrieve semantic keywords in P2P overlay and find extensive service descriptions. When a query with the given key is originated from a peer node in Skip Graph overlay network, ServiceIndex system will compare the given key with the local keys at the current peer node first, then look up its neighbors’ to try to find the peer whose key is closest to the given key, and forward the query message to this neighbor. This process continues until the given key is found, or it reaches the lowest level neighbors and fails.

After completing service inquiry, we realize the functionality matching. We suppose the similarity between elements and attributes are obvious from their names, types and structural grouping of attributes with referring to the same OWL concepts and the associated semantics. Thus we use VSM (Vector Space Model) to computing similarity between query and WSDL-S document or between WSDL-S documents.

\[
       Sim(d_i, q) = \frac{\sum_{j=1}^{n} \omega_{i,j} \times \omega_{i,q}}{|d_i| \times |q|} = \frac{\sum_{j=1}^{n} \omega_{i,j} \times \omega_{i,q}}{|d_i| \times |q|}
\]  
(3)

Where \( \overrightarrow{d_i} \) and \( \overrightarrow{q} \) are the weights in the two vector representations, \( |d_i| \) and \( |q| \) assure that \( 0 \leq Sim(d_i, q) \leq 1 \). The similarity is computed as the cosine of the inner product between their vectors. Function(3) helps to find exact or most similar services for service requesters. Given a query, all documents are ranked according to their similarity with the query.

6. Experimental Evaluation

We develop the ServiceIndex system for service discovery based on Skip Graph P2P overlay network. Unlike
traditional Service-Oriented Architecture, in ServiceIndex system, service providers, service consumers, and service registry behave as peer nodes. We experimented with the real services at Internet. It includes about 3,523 WSDL documents. These documents were labeled with ontologies and transferred into WSDL-S documents using the WSDL2OWL-S tool and manually. We set up a Skip Graph simulator to index segments split from the WSDL-S documents. We evaluated the average query hops (visited nodes during a search), and search time. These two metrics help to reveal the performance of ServiceIndex system.

Table 2 shows the parameters and their initial value in our experiments. We repeated each experiment with 100 times to calculate the average results.

<table>
<thead>
<tr>
<th>description</th>
<th>initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$ number of WSDL-S documents</td>
<td>3,523</td>
</tr>
<tr>
<td>$n$ number of segments or keys</td>
<td>172,627</td>
</tr>
<tr>
<td>$p$ number of peer nodes</td>
<td>172,627</td>
</tr>
<tr>
<td>lcp the longest common prefix</td>
<td>8</td>
</tr>
<tr>
<td>$T$ the maximum keys in a peer node</td>
<td>110</td>
</tr>
</tbody>
</table>

6.1 Query Hops

When these keys are distributed into ServiceIndex, we got average inquiry hops illustrated at Figure 8.

As expected, the average inquiry hops on keys would be $\log(M)$, where $M$ is the number of ServiceBag. That means the inquiry time and messages are correlative with the number of ServiceBag other than the number of keys in our algorithm which would reduce the time of service discovery.

The value of the radius can affect the number of ServiceBag. The larger is the radius, the more keys would be in the same ServiceBag. Figure 9 depicts this trend according to different values of the radius.

As shown in Figure 9, when radius is too small, such as $r = 1.25e^{-5}$, the number of representative peers will increase sharply as the insertion of keys. That means ServiceBags are created frequently which results in “One_ServiceBag-One-Key” and causes great cost of ServiceIndex. When radius is too big, such as $r = 0.7716e^{-3}$, the number of ServiceBag increases placidly with incremental keys, which would cause rapid expansion of ServiceBag. They both will cause the load imbalance of ServiceIndex.

Figure 10 presents the distribution of keys with the radius $2.0e^{-5}$. In this simulation, we first distribute 500 WSDL documents into ServiceIndex and then get about 1,025 ServiceBags. As shown in Figure 10, most ServiceBags are in the 50-80 size range, only 3 ServiceBags are in the 1-10 range, and 2 ServiceBags are larger than 100.
6.2 Search Time

Search time reveals the efficiency of responding to the consumers’ inquiry. Figure 11 shows the comparison of the search time between one-layer Skip Graph and two-layer Skip Graph. We can see that the average search time is about 1.6s-2s in two-layer Skip Graph while 2.7s-3s in one-layer Skip Graph. This is because the two-layer Skip Graph aggregates more similar keys in one peer node and thus reduces the scale of the overlay network. For example, there would create 172,627 peer nodes in one-layer Skip Graph while two-layer Skip Graph only generates 4,721 peer nodes when distributing 3,523 WSDL-S documents.

![Graph showing search time comparison](image)

**Figure 11. The Average Query Execute Time**

7. Conclusions and Future Work

Web Service discovery is an important aspect in service oriented technology. We developed the ServiceIndex system for service discovery which merges advantages of P2P computing and Semantic Web Services into Web Services world. The ServiceIndex system tries to solve the problem of semantic search in distributed environment and support complex search, tree lookups, locality sensitivity, and ontology based service discovery.

We did experiments on the performance and accuracy of the ServiceIndex system. The experimental results indicate that it is possible to construct a dynamic and pure P2P overlay network for service discovery and achieve considerable system performance.

Because our P2P discovery mechanism is based on XML-like descriptions, ServiceIndex system can be applied to other xml-based domains with little change, such as NewsML, MathML and so forth.

Still there are several problems should be solved, such as how to split the WSDL-S documents dynamically, how to proportion the distribution of keys in multi-layer Skip Graph, and how to stabilize the Skip Graph when a large amount of service providers leave from the system that brings peer nodes’ fluctuation. In our future work, we would analyze these problems further and do the corresponding experiments for the ServiceIndex system.

### References