ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com
Special Issue: International Conference on Advanced Research Methodology Held by The International Association for Scientific Research & Engineering Technology, India

A New Service Finding Loom On Decentralized Peer-To-Peer Network

D.Sreenivasulu[1]
PG M.Tech[CSE]
Siddartha Institute of Science and Technology, Puttur

P.Balaji[2]
Assistant Professor,
Siddartha Institute of Science and Technology, Puttur

Abstract- The Service-Oriented Computing (SOC) is emerging as a standard for developing distributed applications, having scalable, reliable, and robust service discovery mechanism is a critical issue of utilizing SOC. In traditional service discovery methods for large scalable service networks, centralized registries are used which can suffer from problems like performance bottleneck and vulnerability to failures. A peer-to-peer based decentralized service discovery approach appears to be the most natural way to address the above issues and achieve scalable, reliable and robust service discovery. To alleviate these problems switching from centralized to decentralized service discovery approaches are required. In this paper the technique for switching from centralized to decentralized service using Chord4s structured p2p is presented. The developed technique is compared with different approaches of peer-to-peer-based decentralized service discovery for evaluating the performance. The results obtained from the proposed technique shows that, the developed method is an efficient and effective for higher data availability by availing the decentralized service discovery.

Keywords- Service Discovery, Peer-To-Peer, Chord, Chord4s

I. Introduction

A peer-to-peer (P2P) network is a distributed system in which peers employ distributed resources to perform a critical function in a decentralized fashion. Nodes in a P2P network normally play equal role, therefore, these nodes are also called peers. In a highly structured P2P network such as Chord4s, both the network architecture and the data placement are precisely specified. The neighbors of a node are well-defined. The data is stored in a well defined location. Service computing refers to a flexible computing architecture that packages functionality as a suite of interoperable routines that can be used within multiple different systems from several business domains Loose coupling of services with operating systems and other technologies that underlie applications is required for Service Computing. Functions are distinguished into distinct autonomous and self-describing units, or services, which developers make accessible via pre-defined interfaces over a network in order to allow users to combine and reuse them in the production of applications. These services communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services [1]. By this way, Service-Oriented Computing (SOC) is emerging as a standard for developing distributed application, but having scalable, reliable, and robust service discovery mechanism is a critical issue of utilizing SOC. The Peer-to-Peer (P2P) technology removes centralized infrastructures to provide a universal approach for improving scalability, robustness and reliability of distributed systems. In areas such as file sharing, Voice over Internet Protocol (VoIP) and video streaming, P2P has achieved great success [8]. To leverage P2P computing and web services for improved service discovery, continuous research is going on in the SOC field. In particular, structured P2P systems such as Chord, CAN, and Tapstry have some characteristics that are suitable for facilitating efficient decentralized service discovery.

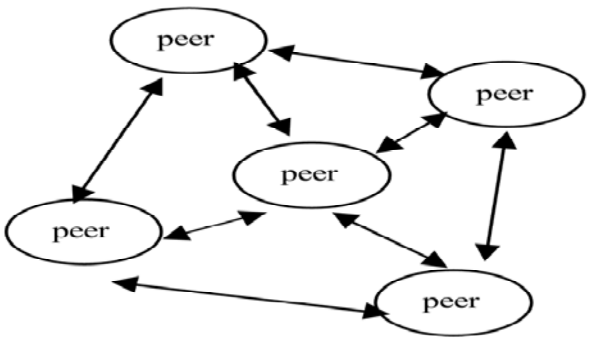


Fig 1. Decentralized Peer-to-Peer File Sharing.

In P2P based decentralized service discovery approach, set of distributed nodes are present which forms the P2P network. When provider registers the service, it is assigned to the relative service node for storing into the repository. Consumer can submit service query to any of the node from the network and if that node does not contain the required service description then it routes the query to respective node. Description of the matched query is then retrieved from the node and returned to the service consumer as a query result. This is the overall idea about how exactly service request is processed in P2P based decentralized service discovery approach. To implement such a service discovery approach DHT based and Chord4s-based approaches are studied. By making use of Distributed Hashing Table (DHT), even data distribution and efficient query routing can be achieved in structured P2P systems. But in DHT based systems, descriptions of functionally equivalent services is distributed on the same successor node because hashing value is similar for these nodes. If such a node fails, any of these services will not be available to the consumer and hence DHT based P2P approaches to decentralized service discovery may not be that efficient in terms of availability of service. This disadvantage may result in serious problems in open and dynamic SOC environments where unexpected failure of nodes cannot be avoided [8]. In Chord4s-based approach, Chord4s has been used to facilitate decentralized web service discovery. Data availability is improved by distributing published descriptions of functionally equivalent services would be stored at the different successor nodes that are organized into virtual segment in the chord4s circle. Another two features of chord4s are to support service discovery with wildcard(s) and QoS awareness.

II. Related Work

In, Hu and Seneviratne propose the approach based on the concept that service providers themselves should take the responsibility to maintain their own service descriptions in a decentralized environment. To group peer nodes by service categories to form islands on the Chord ring, decentralized service directory infrastructure is built with hashing descriptive strings into the identifiers. To handle routing across islands and within islands, Island Table and Native Table are created on every peer node respectively. Schmidt and Parashar in describe a system that supports complex queries containing keywords, partial keywords and wildcards by implementing an Internet-scale DHT. This system assures that all existing data elements matching a query will be returned in terms of number of messages and number of nodes involved. To map the multidimensional information space to physical peers effectively, key innovation, a dimension reducing indexing scheme is used. It provides Chord having ability to perform metric-based similarity search. Node failures would lead to severe data loss when above approaches in [5], [6], [7], [11] are adopted to provide service discovery because descriptions of functionally equivalent services would be stored at the same successor nodes.

III. System Modules

Service description supported by the Chord4s contains our parts those are as follows:

a) Service Description

Service description contains service identifier for identifying the services available in the network. It divides into the two parts, function bits and provider bits. SHA-1 hashing function is using for generating the unique service identifier. Function bits are used to refer to the functionality of the service while provider bits are used to describe the provider specific information.

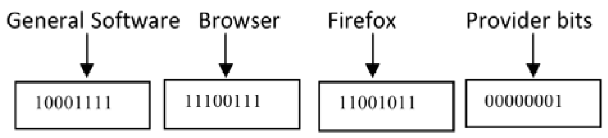


Fig 2. Service identifier generation.

b) Service Publication

Chord4s improves the availability of data by distributing the description of functionally equivalent services among all the peer nodes. The overlay network consisting of n nodes, let m be the length of service identifier and maximum number of functionally equivalent services be k. The length of provider bits x should be carefully calculated to achieve even service description distribution. Then to allocate enough bits for provider bits, it should satisfies the constraint(1) as follows

$$2^x \geq (k-1)2^m/n. \qquad (1)$$

c) Service Query.

Chord4s supports the two types of query: service-specific queries and queries with wildcard(s).In service specific query the service consumer can compose the query with explicit service information, Later it can be hashed using SHA-1 algorithm and the results are used to generate the function bits of the target service identifier. Using the routing information in each peer node query is forwarding from one node to other node until matched service descriptions has to found and those results are sent back to the requested service consumer. But in case of query with wildcard service consumers has to search for the categories of services. Here it provides the good searching results compared to service-specific query. The main advantage of using the query with wildcard is, suppose that composed specific services are not available in the nodes at least it provides the categories of services to the service query composer.

IV. Problem Scope of the Project

Traditional service discovery approaches of the web services technology are based on Universal Description, Discovery, and Integration (UDDI). However, centralized service registries used by UDDI may easily suffer from problems in an open SOC environment. To overcome the problems The Peer-to-Peer (P2P) technology provides a universal approach to improving reliability, scalability, and robustness of distributed systems by removing centralized infrastructures. Based on Distributed Hashing Table (DHT), structured P2P systems can achieve even data distribution and efficient query routing by controlling the topology and imposing constraints on the data distribution. In this technology also problems occur. It is suffered from such as Problems; they are Bottleneck, Vulnerability to failure in large scale network. Large scale service network, largely distributed, unexpected failure of nodes cannot be avoided high cost and loss of control.

V. Architecture

P2P decision tree induction algorithm in which every peer learns and maintains the correct decision tree compared to a centralized scenario. Our algorithm is completely decentralized, asynchronous, and adapts smoothly to changes in the data and the network, shown in Fig.1The client request the service to P2P then collect the dataset and using the ID3 algorithm is efficient in the sense that as long as the decision tree represents the data, the communication overhead is low compared to a broadcastbased algorithm. As a result, the algorithm is highly scalable. When the data distribution changes, the decision tree is updated automatically Our work is the first of its kind in the sense that it induces decision trees in large P2P systems in a communication-efficient manner without the need for global synchronization and the tree is the same that would have been induced given all the data to all the peers.

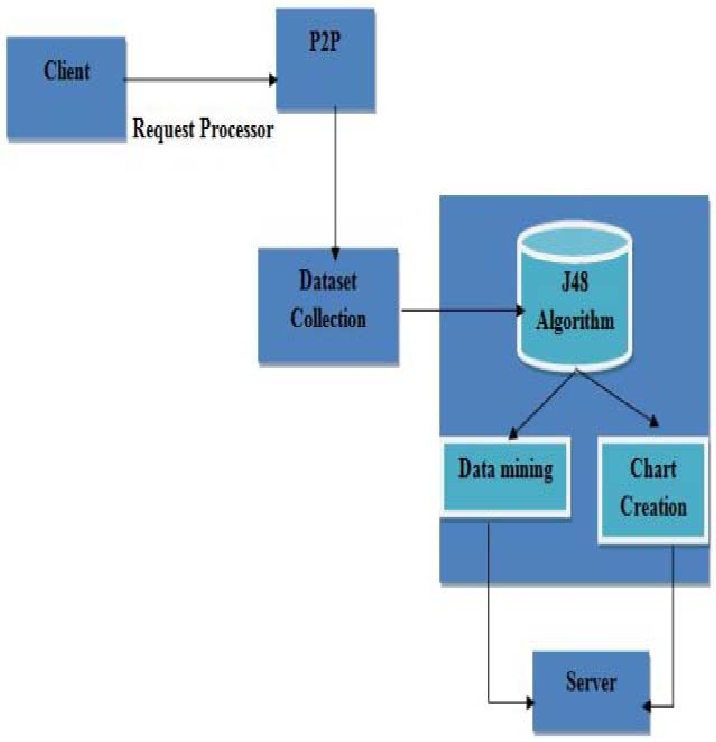


Fig. 3 Decentralized P2P Network in Data mining.

VI. METHODOLOGIES

A collection of related sets of information that is composed a separate elements but can be manipulated as a unit by a computer. Datasets consist of all of the information gathered during a survey which needs to be analyzed. Learning how to interpret the results is a key component to the survey process.

Dataset

0,tcp,auth,SF,9,36,0,0,1,0,0,1,normal

0,tcp,daytime,S0,0,0,0,0,0,0,0,0,249,DOS

0,tcp,auth,SF,10,37,0,0,1,0,0,1,normal

0,tcp,auth,SF,10,38,0,0,1,0,0,1,normal

0,tcp,daytime,S0,0,0,0,0,0,0,0,0,252,DOS

0,tcp,daytime,S0,0,0,0,0,0,0,0,0,217,probe

0,tcp,daytime,SH,0,0,0,0,0,0,0,0,1,probe

0,tcp,daytime,S0,0,0,0,0,0,0,0,0,245,DOS

0,tcp,bgp,S0,0,0,0,0,0,0,0,0,249,DOS

0,tcp,bgp,S0,0,0,0,0,0,0,0,0,250,DOS

0,tcp,bgp,S0,0,0,0,0,0,0,0,0,253,DOS

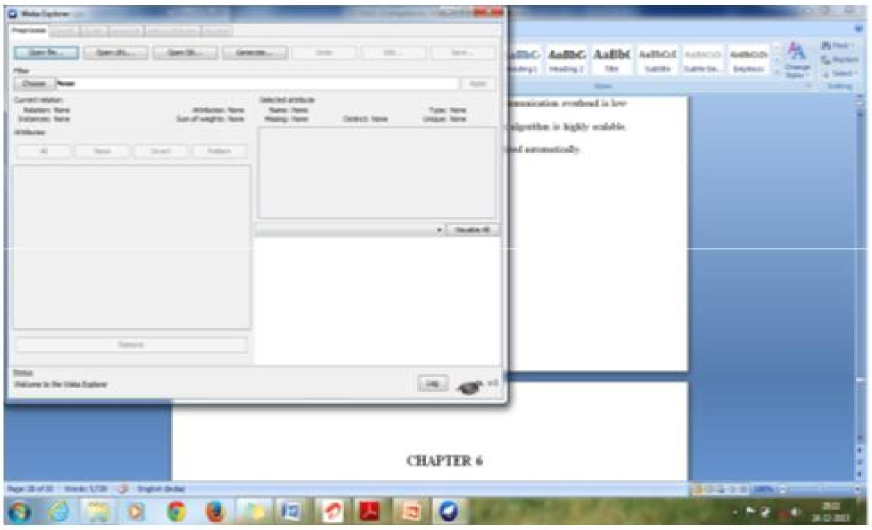


Fig. 4 Open Weka

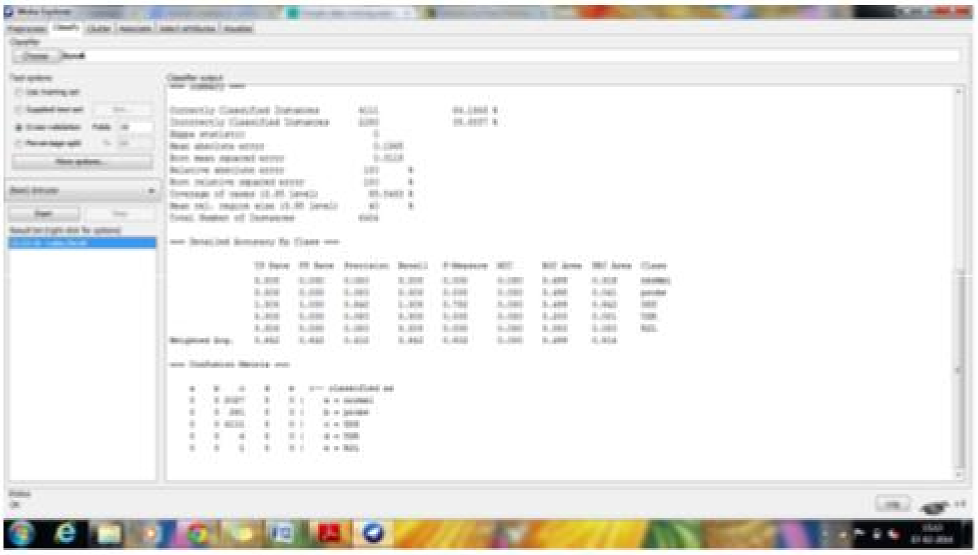
J48 [QUI93] implements Quinlan□s C4.5 algorithm [QUI92] for generating a pruned or un pruned C4.5 decision tree. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by J48 can be used for classification. J48 builds decision trees from a set of labeled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree telling to choose that class. But it can also happen that none of the features give any information gain. In this case J48 creates a decision node higher up in the tree using the expected value of the class. J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. Further it provides an option for pruning trees after creation.

Program:

```
package weka.classifiers.trees;

import weka.classifiers.AbstractClassifierTest;

import weka.classifiers.Classifier;

import junit.framework.Test;

import junit.framework.TestSuite;

public class J48Test extends AbstractClassifierTest {

public J48Test(String name)

{ super(name); } /**

Creates a default J48 */

public Classifier getClassifier()

{

return new J48();

}
public static Test suite()

{

return new TestSuite(J48Test.class);

}

public static void main(String[] args)

{

junit.textui.TestRunner.run(suite());

}

}
```

Resource Allocation:

Resource allocation is the scheduling of activities and the resources required by those activities while taking into consideration both the resource availability and the project time. Cost of resources varies significantly depending on configuration for using them. The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.



```
=== Summary ===

Correctly Classified Instances        [illegible]        [illegible] %
Incorrectly Classified Instances      [illegible]        [illegible] %
Kappa statistic                         [illegible]
Mean absolute error                     [illegible]
Root mean squared error                 [illegible]
Relative absolute error               [illegible]   %
Root relative squared error           [illegible]   %
Coverage of cases (0.95 level)        [illegible] %
Mean rel. region size (0.95 level)    [illegible]   %
Total Number of Instances             [illegible]

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 [illegible]
Weighted Avg.    [illegible]

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
    [illegible]
```

Fig. 5 Cross-validation test

VII. SERVICE DISCOVERY

Searching in strictly structured P2Ps The searching algorithms support two basic operations: lookup(key) and put(key). lookup(k) is used to find the location of the node that is responsible for the key k. put(k) is used to store a data item (or a pointer to the data item) with the key k in the node responsible for k. In a distributed storage application using a DHT, a node must publish the files that are originally stored on it before these files can be retrieved by other nodes. A file is published using put(k).

Searching in non-hierarchical DHT P2Ps Different non-hierarchical DHT P2Ps use different flat data structures to implement the DHT. These flat data structures include ring, mesh, hypercube. Chord [12] uses a ring data structure. Node IDs form a ring. Each node keeps a finger table that contains the IP addresses of nodes that are half of the ID ring away from it, one fourth of the ID ring away, one-eighth of the ID ring away, until its immediate successor. A key is mapped to a node whose ID is the largest number which does not exceed that key. During the searching for lookup(k), a node A forwards the query for k to successor(k), which is another node in A's finger table with the highest ID that is not larger than k. In this way, the query for k is forwarded through the successor list until the node responsible for k is reached. The finger table speeds up the lookup operation. In case of the failure of successor(k), a node forwards the query to its immediate successor node. Chord achieves O(log N) routing efficiency at the cost of O(log N) routing state per node. N refers to the total number of nodes in the system. The work in [12] extends Chord by adding different kinds of reverse edges into Chord so that the modified Chord is resilient to routing attacks.

In Chord4s [8] any piece of information has to be uniquely identifiable by a key. This is achieved by applying a hash function on any data which is to be stored in the network. This key is used for storage as well as for retrieval of data. Consequently, in order to find any information the full key has to be known in advance assuming that one has to know exactly what to look for. Searching capability has to be implemented separately of this lookup mechanism or by using additional data structures, e.g. inverted indices. All nodes in the overlay network are assigned unique identifiers (ID) of the same key space. Any node is responsible for the data keys within the range of the next smaller node ID in the network up to its own node ID. Therefore every node has to know its predecessor which is propagated and updated by maintenance messages. The ordering of nodes in successor-predecessor relations forms the so-called Chord4s ring. In addition to predecessor references every node stores a so-called finger table—a skip list containing i references to nodes of which the node IDs are at least the i-th power of two greater than their own node ID. By this, queries can be forwarded at least half the way closer to their destination in every step only needing to know a logarithmic number of nodes in the network. This leads to logarithmic performance for storing and retrieving any item stored in the ring. Whenever nodes join or leave the network routing information have to be updated. This is done by a stabilization protocol which has been proven to be correct even in case of multiple nodes joins or leaves at the same time [8]. In order to prevent data loss because of node failures data should be replicated on multiple nodes in a network, e.g. by copying it on the k next nodes following a node responsible for a specific data item.

VIII. EXPERIMENTAL RESULTS

Chord4s protocol is using for the service description distribution and discovery. It organizes the nodes in the form of ring as shown in fig 3. First we have to set the services among all the peer nodes, then go to system administration part and uploading the services among all the peer nodes. Get the conformation about service updating. User can select any one of the service specific query, it search in ring format one after the other. Service is downloaded from the particular peer node. After that we can search the query with wildcard it take more time than the previous one because here it has search the similarly service and avoid failure of the query. While transmitting the services from one node to other node we are using the Advanced Encryption Standard algorithm for providing security

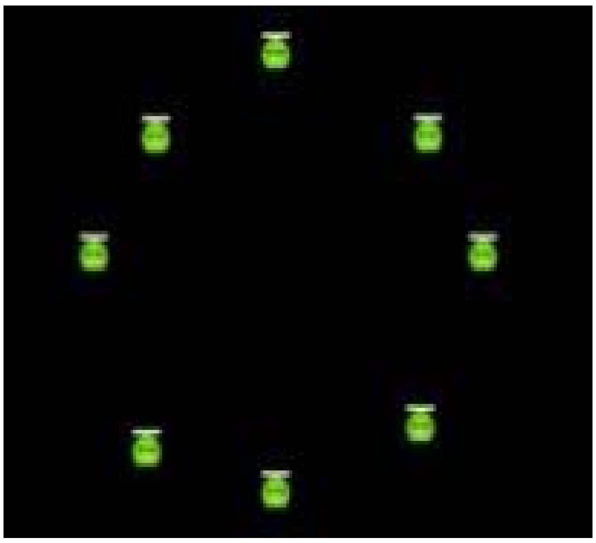


Fig 6.Network Creation.

After performing experiment in the above Fig 4 it clearly shows that our proposed method is good compared to exiting one because it reduces the number of queries are going to fail. In the network suppose any one of the node fails due to some reasons that query should be handled by the other node which is existing in the network. But this is not possible in the chord method if any node fails the entire information stored in that is completely loss as result it increase the number of queries to be failed. In Fig 4 the above line shows the chord method in which failure of the query rate is high and the below line shows proposed method in which failure of the query rate is low .finally we conclude that data availability is good in the proposed method.

Here conducting the experiment on service discovery approach. In Fig 5 it shows comparison between the service discovery with the specific query and query with wild card. The service discovery with wildcard query gives good searching results compared with specific one. While in specific query it search only for the accurate service but in query with wildcard it search for the category for services. In the Fig 5. The above line shows the query with wildcard, it has to search the category of services because of that it visit the large number of nodes, enough the requested query is not existing in the network it give the similar service which is available in existing network .The below line shows that specific query. Suppose that requested query is not available in the network directly it will send the response to the query composer as no similar services are found. In this way the proposed method is more efficient compared to the existing one.

IX. Conclusion

In service-oriented computing service discovery is challenging task. We are achieving this one by using the chord4s protocol, it distribute the description of functionally equivalent services among all the peer nodes.Chord4s supports the QOS -aware service discovery and service discovery with wildcard(s). Experimental description demonstrate that Chord4s achieve high data availability and providing the security by using the Advanced Encryption Standard algorithm.

References

1. R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba, "Resource and Service Discovery in Large-Scale Multi-Domain Networks," IEEE Comm. Surveys and Tutorials, vol. 9, no. 4, pp. 2-30, Oct.-Dec. 2007.
2. E. Al-Masri and Q.H. Mahmoud, "Crawling Multiple UDDI Business Registries," Proc. 16th Int'l Conf. World Wide Web (WWW '07), pp. 1255-1256, 2007.
3. D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani, "PAWS: A Framework for Executing Adaptive Web-Service Processes," IEEE Software, vol. 24, no. 6, pp. 39-46, Nov./Dec. 2007.
4. J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. St.John, J. Schlimmer, G. Simonet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri, "Web Services Dynamic Discovery (WS-Discovery)," http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf, 2005.
5. F. Emekc¸i, O.D. Sahin, D. Agrawal, and A.E. Abbadi, "A Peer-to- Peer Framework for Web Service Discovery with Ranking," Proc. IEEE Int'l Conf. Web Services (ICWS '04), pp. 192-199, 2004.
6. T.H.-T. Hu and A. Seneviratne, "Autonomic Peer-to-Peer Service Directory," IEICE Trans. Information System, vol. E88-D, no. 12, pp. 2630-2639, 2005.
7. Y. Li, F. Zou, Z. Wu, and F. Ma, "PWSD: A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network," Proc. Sixth Asia-Pacific Web Conf. Advanced Web Technologies and Applications (APWeb '04), pp. 291-300, 2004.
8. Qiang He, Member, IEEE, Jun Yan, Yun Yang, Ryszard Kowalczyk, and Hai Jin, Senior Member, IEEE, "A Decentralized Service Discovery Approach on Peer-to-Peer Networks", IEEE Transaction on Services Computing, VOL. 6, NO. 1, JANUARY-MARCH 2013.
9. A.I.T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01), pp. 329-350, 2001.
10. B. Sapkota, D. Roman, S.R. Kruk, and D. Fensel, "Distributed Web Service Discovery Architecture," Proc. Advanced Int'l Conf. Telecomm. and Int'l Conf. Internet and Web Applications and Services, p. 136, 2006.
11. C. Schmidt and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," World Wide Web, vol. 7, no. 2, pp. 211-229, 2004.
12. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '01), pp. 149-160, 2001.