*International Journal of Research in Engineering and Science (IJRES)*
*ISSN (Online): 2320-9364, ISSN (Print): 2320-9356*
*www.ijres.org Volume 2 Issue 6 ǁ June. 2014 ǁ PP.01-07*

# An Enhanced P2P Architecture for Dispersed Service Discovery

M. Divya, A. Anbumani
*Department of Computer & Communication Mahendra Institute of Technology, Tiruchencode,Tamilnadu.*
*Department of Computer & Communication Mahendra Institute of Technology, Tiruchencode, Tamilnadu.*

***Abstract***
Service discovery is a critical issue in Service Oriented Computing (SOC).Service discovery protocols used to detect and discover services offered by the nodes in the network. It must be scalable, reliable and robust service discovery mechanism. In traditional discovery mechanism uses decentralized service discovery approach named as chord4s.This method suffer from some problems such as scalability, node failure and efficient query routing. This paper addresses additional functionalities of chord4s protocol. In this paper data availability in chord4s protocol is improved by distributing functionally equivalent services to different successor nodes. If one node fails service consumer gets equivalent services from the other successor nodes. In this paper efficient query routing can be improved by getting multiple services with single query. Quality of service also improved by qos aware service discovery methods. Semantic information of services is integrated in order to increase flexibility, accuracy of service discovery.
***Index Terms***- Service Oriented Computing, Chord4s

## I. INTRODUCTION

Web services are becoming an emerging technology for developing distributed applications. Service discovery is an important issue in service oriented computing (SOC).Service Discovery protocols are used to detect and discover services offered by the nodes in a network. SOC is being adopted by major computer uses, including banks (Web banking services), retailers (Web shopping services), and airlines (Web booking services).

Traditional service discovery approaches based on UDDI Business Registries(UBR).This registries uses Centralized service discovery approach. This method uses Web Service Crawler Engine (WSCE) [2] to discover web services in effective manner. This method is not suitable for large scale distributed environment. It produces many problems such as performance bottleneck and vulnerability to failures.

Number of service consumers and requests increase in an open SOC environment. SOC is a large scale distributed environment. Centralized registries are not suitable for large scale networks. To avoid these problems decentralized service discovery approach is used to discover and distribute services in large scale, distributed SOC environment. In

Centralized service discovery approach-h each and every service request and response depends on that centralized registries only. In decentralized service discovery approach each and every service request and response does not depends on the centralized registries. Each and every node in the network shares its details with other nodes. The Peer-to-Peer (P2P) technology is a type of decentralized and distributed network. In that individual nodes in the peers act as both service consumers and service providers. This method is a universal approach to improving reliability, scalability, and robustness of distributed, systems by removing centralized infrastructures. Emerging applications of peer to peer technologies are content distribution, file sharing, web search engines and real time communication like skype[3]. Service discovery method uses some of the Structured P2P systems such as Chord [21], CAN [16],Pastry[18],andTapstry[ 23]

In this paper uses chord4s protocol for discover services in decentralized manner. Chord4s is suitable for large scale distributed networks. Chord4s is selected for its scalability. Traditional service discovery approach each and every request depends on UBR central registry. If no of request is increased, response from the central registry is decreased. Service consumer cannot get required service.

In decentralized service discovery approach each and every request does not depends on the other nodes Service descriptions are distributed in the peers in networks. Each and every node is having all information about that services which can provided by the service providers. Service query can be submitted by the service consumer it is distributed in the network by using distributed hash function. It can produce hash value for each service descriptions; this hash value is stored in its repository.

Query submitted by the service consumer is compared with the information stored in the repository. If it is matched that particular information is given to the service consumer otherwise that query can be routed into other distributed nodes. In that functionally equivalent services are distributed in different successor nodes. If one node fails service consumer gets equivalent services from the other successor nodes .It improves availability

of data to the service consumers. This method produces efficient query routing. Service consumer can get multiple services with single query. Service query from the consumer is segregated into three main parts that is service identifiers, qos specification and syntax specification. Service identifiers are further divided into two parts that is functional bits and provider bits. Using this provider bits service provider can provide multiple services with single query. In Chord4s semantic information of services is integrated in order to increase flexibility, accuracy of service discovery.

The rest of the paper is organized as follows. Section 2 introduces major related work. Section 3 presents the unique service description of Chord4S. Then, Section 4 addresses the service publication approach. After that, the new routing protocol of Chord4S for service discovery is proposed in Section 5, followed by discussion of experimental results in Section 6. Finally, Section 7 summarizes the major contribution of this paper and outlines future work.

## II. RELATED WORKS

This section briefly summarizes related work. Our research aims at providing a scalable, reliable, and robust approach for web service discovery.

***Traditional Service Discovery Method***

Traditional service discovery approaches based on UDDI Business Registries (UBR)[2].This registries uses Centralized service discovery approach. This method uses Web Service Crawler Engine to discover web services in effective manner. This method is not suitable for large scale distributed environment. It produces many problems such as performance bottleneck and vulnerability to failures. Number of service consumers and requests increase in an open SOC environment. SOC is a large scale distributed environment. Centralized registries are not suitable for large scale networks. To avoid these problem Decentralized service discovery approach is used to discover and distribute services in large scale, distributed SOC environment.

***Decentralized* Service Discoveries**

Traditional decentralized approach uses chord protocol which is a distributed hash table protocol. This protocol is well suitable for peer to peer networks which effectively locate the node that stores the particular data items. Joining of new nodes and leaving of old nodes can be effectively performed in the peer to peer circle in the network by using chord protocol. It needs routing of information about the other nodes.

Finger table is used to store the routing information about the other nodes. It maintains the details about predecessor and successor nodes. The node with the highest identifier less than n's identifier, allowing for wraparound is called predecessor. The node with the lowest identifier greater than n's identifier, allowing for wraparound is called successor nodes. Finding predecessor and successor nodes are main critical issue in chord protocol. So, this method is not So, this method is not suitable for large scale distributed environment.

One of the structured p2p systems Content Addressable network (CAN) is used for peer to peer file sharing. The CAN is scalable, fault tolerance and completely self organizing. It is used in large scale storage management systems. This system requires efficient insertion and retrieval of content in large scale systems. Departure of the nodes and recovery is the main issue in the construction of CAN[16]. Multiple hash functions are used in CAN. It produces replicated data items

Li et al. [13] present PSWD, a distributed web service discovery architecture based on an extended Chord algorithm called XChord. .In this paper XML is used to web service description and service query. Schmidt and Parashar [20] describe a system that implements an Internet-scale DHT .This system supports search using keys and partial key .M-Chord [15] provides Chord with the ability to perform metric-based similarity search

## III. SERVICE DESCRIPTION

Chord4s protocol describes how to distribute and discover services in decentralized manner, query routing and how to provide quality of services to the service consumers.

***Chord4s Implementation***

Chord4s protocol supports one operation; given a key, it will determine the node responsible for storing the key value. It uses consistent hashing which is used to assign hash keys to the nodes. It defines the node responsible for a key to be that keys successor. The node with the lowest identifier greater than n's identifier, allowing for wraparound is called successor nodes. Finding successor nodes is primary task in chord4s protocol.Basic operations used in the chord4s protocol are insert(key,value),lookup(key),join(n) and leave(n)

***Service Descriptions***

The service description supported by Chord4S consists of three main parts: service identifier, QoS specification, and syntax specification. The service identifiers are used for routing query messages. Service identifiers consist of provider bits and functional bits .Provider bits are used to provide information and functional bits are used to refer the functionality of services.Chord4S supports distribution and query for hierarchical service description, e.g., "Booking. Hotel.America.USA.Texas.Huston" and "Multimedia. Video.Encoder.AVI2RM." Service consumer sends this query to the service provider, the description of the service is stored in its repository. Service identifiers are generated to route the query. Functional bits and provider bits are generated depending upon the functions and behavior of the query. Depending upon the provider bits the functionally equivalent services are stored in the same successor nodes. If one node fails service consumer gets equivalent services from the other successor nodes

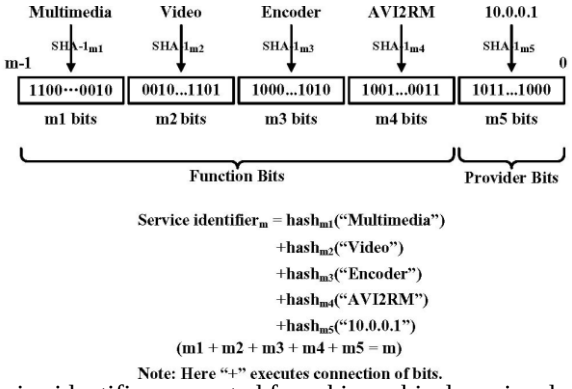



Fig 1: Service identifier generated from hierarchical service descriptions

A sample of service identifier consisting of five layers is presented in Fig. 1. The function bits are used for the functional service matchmaking in service discovery. The main function of the provider bits is to distinguish and distribute functionally equivalent services. Using SHA-1[21]one of the general consistent hashing functions, the probability of hashing two service descriptions to a same value is negligible as long as the length of the provider bits is large enough. Identifiers generated from functionally equivalent services differ from each other in a certain number of the lowest bits, i.e., the provider bits. For example, in an application with a maximum of five layered (four layers for functional bits and one layer for provider bits) service description, a service description like "Multimedia.Video.AVIPlayer" is also acceptable. When generating service identifier for this service description, the first three layers of the service identifier will be generated by using hashm1 ("Multimedia"), hashm2 ("Video"), and hashm3 ("AVIPlayer"). The fourth layer would be zero by default. In this case, the service description will be placed in a virtual segment containing all the service descriptions starting with "Multimedia.Video."

***QoS Specification***

QoS awareness is a critical issue in an SOC environment. QoS-aware service discovery used to provide quality of service to the service consumers.It only returns the service which will meet the requirements of the service consumer. Chord4S allows service providers to publish their services with quality specifications attached as advertisements. However, the quality specifications are not involved in the generation of service identifier. After finding a service description that matches its functional requirements according to the service identifier, the service consumer can look over the attached quality specification. Chord4S supports three types of QoS attributes, defined as follows:

**1. Numeric QoS attributes**, e.g., <; >;<¼;> ¼ etc, are often used to specify service consumer's QoS requirements of this type, such as "Price < ¼ $1; 000:00" and "Availability > ¼ 0:95".

2**. Boolean QoS attributes**. A Boolean QoS attribute is a QoS attribute that can be assigned with one of the two values: true and false. Two comparison operators, ¼¼ and ! ¼ can be used to specify QoS requirements of this type, such as "Cancellable = = True."

**3. Enumerated type**. An enumerated type of QoS attribute is a QoS attribute that can be assigned with any of the enumerators as a value. For example, the types of an international postal service can be declared an enumerated type of QoS attribute that can be assigned with one of the three enumerators: Airmail, Registered Mail, or Express Mail. Two comparison operators, ¼¼ and ! ¼, as well as set operators can be used to specify

QoS requirements of this type, such as "Type = = Registered Mail" and "Type C_ {Registered Mail, Express Mail}." For example, a service consumer that requires an registered or express mail to be delivered at a price no more than $100.00 can specify a combination of QoS requirements as "Type _ {Registered Mail, Express Mail} AND Price < ¼ $100:00. Besides QoS specifications, other service-specific information can be published by service providers, e.g. the behavior of services in the context, such as how a web service is used in a business process and how services interact with each other in a service composition scenario. This kind of information can be taken into consideration when looking up service providers in complicated applications. To name a few, BPEL and OWL-S descriptions can be converted to finite automata through several methods [5], [8], [14]. Then the results from hashing the finite automata or the Path Finite Automata (PFA) generated from the finite automata can be incorporated into the service description to enable semantic-enhanced service discovery [7].

## IV. SERVICE PUBLICATIONS

In this section, we present the mechanisms for distributed service publication.

***Traditional Approach in Chord***

Traditional decentralized approach uses chord protocol which is a distributed hash table protocol. This protocol is well suitable for peer to peer networks which effectively locate the node that stores the particular data items. Joining of new nodes and leaving of old nodes can be effectively performed in the peer to peer circle in the network by using chord protocol. It needs routing of information about the other nodes. Finger table is used to store the routing information about the other nodes. It maintains the details about predecessor and successor nodes. The node with the highest identifier less than n's identifier, allowing for wraparound is called predecessor. The node with the lowest identifier greater than n's identifier, allowing for wraparound is called successor nodes

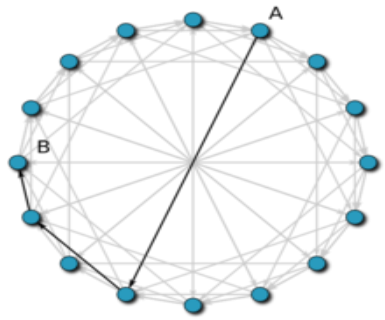



Fig 2: Routing path between A and B

Chord is used to find locations of the key, joining new nodes in a network and used to recover from failure and departure of existing nodes. Hash function assigns each node and key an m-bit identifier using a base hash function such as SHA-1

ID(node) = hash(IP, Port)

ID(key) = hash(key)

Forexample"Multimedia.Video.Encoder.AVI2RM." When these service descriptions are hashed, the returned dentifiers will be the same. Hence, these service descriptions will be stored at the same successor node. Similar to single point failure, failure of this successor node will lead to inaccessibility of all the services of "Multimedia.Video.Encoder.AVI2RM."

***Service Publication in Chord4S***

Two approaches are available in chord4s service publication. Replication and Redundancy storage of multiple copies of a service description at different nodes is called as replication. Storage of redundant information long with the service description is called as redundancy. The replication approach needs to maintain data availability. The redundancy approach requires significant change to the original service descriptions which may not be acceptable by the service providers. Both approaches may result in a considerably large burden on the system. Chord4S improves data availability by distributing descriptions of functionally equivalent services to different nodes. In this way, a failed node would just have limited impact on data availability. A service consumer has the opportunity to locate the functionally equivalent services from those available nodes.

## V. SERVICE QUERY

This section presents how routing of query messages is performed in Chord4S based on the service publication approach ***Query Types***

Chord4S supports two types of query: service-specific queries and queries with wildcard(s).

***Service-Specific Query***

A service-specific query contains complete details of a service description and is used to look up a specific service. In a system that allows four-layered function bits in the service descriptions, "Multimedia.Video.Encoder.AVI2RM" is a typical example of service-specific query

***Query with Wildcard(s)***

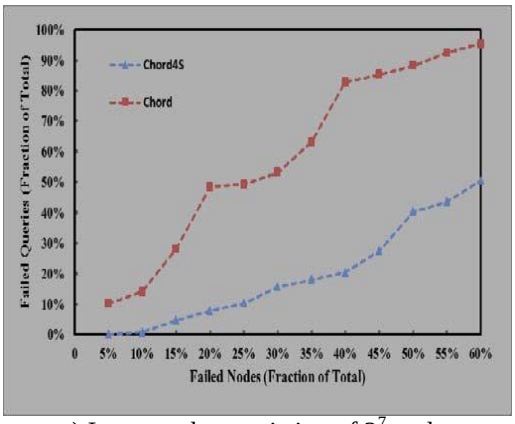
Sometimes service consumers need to search for categories of services. For example, an amplifier service that amplifies the audio of an RM movie file can be composed using three component services which correspond to three specific steps: audio extraction, audio amplification, and video/audio combination. Therefore, the service consumer needs to find the component services from three categories: "Multimedia. Video.AudioExtractor," Multimedia.Audio.Amplifier," and "Multimedia.Combiner," and select the ones whose inputs and outputs match. In such cases, service queries using wildcard(s) are necessary, e.g., Multimedia.Video. AudioExtractor.*", "Multimedia. Audio.Amplifier.*", and Multimedia.Combiner.*.*". When solving a query with wildcard(s), it is actually looking up a virtual segment composed by nodes succeeding service descriptions that fall into the target service category. The generation of target service identifier or more specifically target service category identifier—for a query with wildcard(s) is similar to that for a service specific query. The difference is that the layers corresponding to the wildcard(s) will be stuffed with 0s.

## VI. EXPERIMENTAL EVALUATIONS

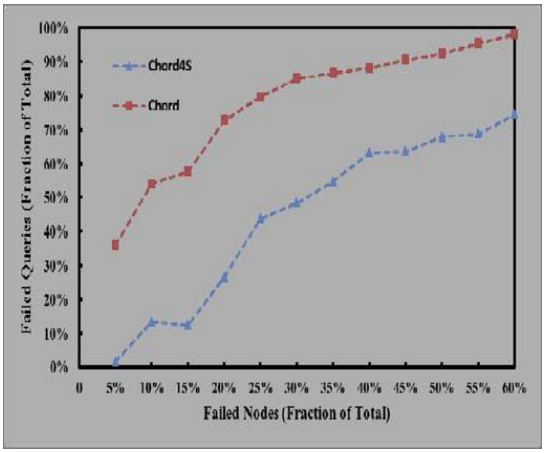
To evaluate the performance of Chord4S, a Chord simulator is extended to support Chord4S topology control, data distribution and routing protocol

***Data Availability***

A unique feature, and also a main design goal of Chord4S is the high data availability in volatile environments. To set up the volatile environments, we randomly select a fraction of nodes participating in the network to fail in each experiment increasing from 5 to 60 percent by steps of 5 percent. Then the remaining nodes were randomly selected to send queries for services. The number of required functionally equivalent services in each query is randomly picked from the interval [1], [12]. To evaluate the data availability, we measured the fraction of the failed queries. We conducted two sets of experiments, one with Chord and the other with Chord4S. Fig.4 compares the results from the two sets of experiments, where the Chord4S curves always start at a much lower point and continue to stay at lower points compared to Chord. It can be observed that the fractions of ailed queries are higher than the fraction of failed nodes. This result indicates that other than the service descriptions loss along the failed nodes something else also caused failed queries. The reason is that without stabilization some entries in existing nodes' finger tables became invalid. Those invalid entries yielded some failed queries and decreased the data availability in both cases of Chord and Chord4S because some queries could not be forwarded correctly. To conclude, the experimental results demonstrate that Chord4S provides better data availability than Chord in different volatile environments on different scales




a) In networks consisting of $2^7$ nodes




b) In networks consisting of $2^{11}$ nodes

Fig 4 Data Availability

## VII. CONCLUSION

Chord4s which is the P2P based decentralized service discovery approach that utilizes data distribution and lookup capabilities of Chord to distribute and discover services in a decentralized manner. Data availability is improved by distributing descriptions of functional equivalent services to different successor nodes It is extended to support efficient discovery of multiple services with single query. Thus Chord4s achieves higher data availability and efficient query with reasonable overhead. Semantic information of services is integrated in chord4s to increase flexibility, accuracy of service discovery. In the future, integration of semantic information of services into Chord4s using popular tools such as Petri net, WSMO to improve efficiency and reduce overhead.

## REFERENCES

[1] "North American Industrial Classification Scheme (NAICS) codes," http://www.naics.com/, 2012.
[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] S. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to- Peer Internet Telephony Protocol," Proc. IEEE INFOCOM, pp. 1- 11, 2006.
[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] Z. Cheng, M.P. Singh, and M.A. Vouk, "Verifying Constraints on Web Service Compositions," Real World Semantic Web Applications, June 2002.
[6] L. Clement, A. Hately, C. von Riegen, and T. Rogers, "UDDI Version 3.0.2," OASIS, http://www.uddi.org/pubs/uddi_v3.htm, 2004.
[7] 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.
[8] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-Based Verification of Web Service Compositions," Proc. IEEE 18th Int'l Conf. Automated Software Eng. (ASE '03), pp. 152-163, 2003.
[9] V. Gopalakrishnan, B.D. Silaghi, B. Bhattacharjee, and P.J. Keleher, "Adaptive Replication in Peer-to-Peer Systems," Proc. 24th Int'l Conf. Distributed Computing Systems (ICDCS '04), pp. 360-369, 2004.
[10] 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.
[11] H.V. Jagadish, B.C. Ooi, K.-L.Y. Tan, and R. Cui Zhang, "iDistance: An Adaptive B+-Tree Based Indexing Method for Nearest Neighbor Search," ACM Trans. Database Systems, vol. 30, no. 2, pp. 364-397, 2005.
[12] L.-j. Jin, V. Machiraju, and A. Sahai, "Analysis on Service Level Agreement of Web Services," technical report, HP Laboratories, http://www.hpl.hp.co.uk/techreports/2002/HPL-2002-180.pdf. 2002.
[13] 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.
[14] S. Narayanan and S.A. McIlraith, "Simulation, Verification and Automated Composition of Web Services," Proc. 11th Int'l Conf. World Wide Web (WWW '02), pp. 77-88, 2002.
[15] D. Novak and P. Zezula, "M-Chord: A Scalable Distributed Similarity Search Structure," Proc. First Int'l Conf. Scalable Information Systems, 2006.
[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. Conf. SIGCOMM, pp. 161-172, 2001.
[17] P. Rompothong and T. Senivongse, "A Query Federation of UDDI Registries," Proc. First Int'l Symp. Information and Comm. Technologies, pp. 561-566, 2003.
[18] 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.
[19] 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.

[20] 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.

[21] 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.

[22] L. Wu, Y. He, D. Wu, and J. Cui, "A Novel Interoperable Model of Distributed UDDI," Proc Int'l Conf. Networking, Architecture, and Storage (NAS '08), pp. 153-154, 2008. COMPUTING, VOL. 6, NO. 1, JANUARY-

[23] B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph, "Tapstry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Computer Science Division of Univ. California, 2001.