Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 9, September -2017

SCALABLE AND RELIABLE APPROACH FOR HIGH DATA AVAILABILITY IN SERVICE ORIENTED COMPUTING ENVIRONMENT AND RESULT ANALYSIS

[1]Neha R. kshirsagar [2]Prof Mrs.V.R.Chirchi

[1]M.*E Student, Department of CNE, M.B.E. Society's College of Engineering, Village Morewadi, Ambajogai, District Beed, Maharashtra, India,*
[2]*Professor, Department of CNE, M.B.E. Society's College of Engineering, Village Morewadi, Ambajogai, District Beed, Maharashtra, India,*

Abstract:- *Service-Oriented Computing (SOC) is increasing as a accepted for developing distributed functions, having scalable, reliable, and sturdy service discovery mechanism could also be a significant issue of utilizing SOC. In recent years, the service discovery ways in which for large scalable service system, centralized collections are used this can suffer from issues like performance susceptibility and bottleneck or traffic jam to failures. We introduced and proposed a peer-to-peer based decentralized service discovery paths view to be the foremost natural because of address the higher than issues and accomplish scalable, reliable and durable service discovery. This paper has proposed Chord4S, a peer-to-peer based approach for decentralized service discovery. To improve data availability, Chord4S distributes the descriptions of functionally equivalent services. Chord4S supports QoS aware service discovery and service discovery with wildcard(s).*

Keywords- *peer-to-peer network, decentralized approach,Chord4S, Search process*

I. INTRODUCTION

Service-Oriented Computing (SOC) is rising as an everyday for developing distributed application, but having scalable, reliable and powerful service discovery mechanism may well be a crucial issue of utilizing SOC. Existing in service discovery path of the web services technology are supported Universal Description, Discovery, and Integration (UDDI) [2]. In ancient service discovery methods for large scalable service networks, centralized registries are used which can suffer from issues like performance bottleneck and vulnerability to failures as a result of sizable quantity of service customers and requests in an open SOC atmosphere. Due to this disadvantage, applying web services in giant scalable service is sometimes prevented the Peer-to-Peer (P2P) technology removes centralized infrastructures to supply a universal approach for rising scalability, robustness and reliability of distributed systems. The advantage of P2P computing and internet services for improved service discovery, continuous analysis goes on within the SOC field. Above all, structured P2P systems such as Chord[3], CAN[4], Pastry[5], and Tapestry[6] have some characteristics that square measure applicable for facilitating economical decentralized service discovery. This paper proposes Chord4S[1], a Chord-based decentralized service discovery approach that supports service description distribution and discovery in a P2P manner.

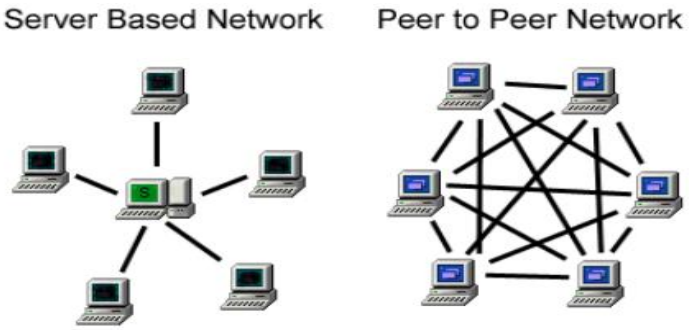


Figure.1: centralized server primarily based and Peer-to-Peer (P2P) networks

In Section II, a comprehensive literature review of existing research is provided. In Section III, of this proposed work of decentralized service discovery on peer-to-peer network. In Section IV**,** Service description support by Chord4S present. In Section V, Service Publication of Chord4S present. In Section VI, Service question with specific query and wildcard(s) present. Also pseudocode for successor node and matchmaking function. In Section VII, Experimental evaluation, routing performance graph present. In Section VIII, summarizes the major contribution of this paper and future scope.

II. LITERATURE REVIEW

2004: V. Gopalakrishnan, B.D. Silaghi, B. Bhattacharjee, and P.J. Keleher researched adaptive replication in Peer-to-Peer Systems**.** Peer-to-peer (P2P) systems perform any number of different functions, but their most fundamental task is that of locating data. This paper describes and characterizes a lightweight, adaptive, system-neutral replication protocol (LAR) that is efficient at redistributing load in such circumstances, and which improves query latency and reliability as well. For purposes of this paper, we define a P2P system as a distributed system where functionally identical servers export data items to each other [7].
2006: Salman A. Baset and Henning Schulzrinne introduced Skype is a peer-to-peer VoIP client allows its users to place voice calls and send text messages to other users of Skype clients. It has central login server. In during experiments, this paper did not observe any subsequent exchange of information with the login server after a user logged onto the Skype network [8].
2007: Service and resource discovery is a crucial research topic, since it is required to support any service infrastructure. It requires the open, scalable, robust and efficient to a greater extent than a single-domain system. This paper survey a number of service discovery approaches and extract those characteristics that make them suitable for this kind of infrastructure Tapestry root node may be a single point of failure. The other nodes must have prior knowledge to allow them to identify the root node [9].

III. PROPOSED WORK

This paper proposes Chord4S[1], a Chord-based decentralized service discovery approach that supports service description distribution and discovery in a P2P manner. Chord is selected because it is well recognized for its flexibility and scalability and is considered suitable in large scale SOC environments. The main aim of designing Chord4S is to largely improve the availability of service descriptions in volatile environments by distributing descriptions of functionally equivalent services to different successor nodes. In case one node fails, a service consumer is still able to find functionally equivalent services that are stored at other successor nodes. Another two features of Chord4S are to support service discovery with wildcard(s) and QoS awareness. Following fig. shows that data flow diagram for proposed work. Admin produce node with service name, ip address and port variety. Consumer searches the query. If question actual match with information (service description) then consumer can transfer it, otherwise wildcard match are used for approximate or near about matching purpose.

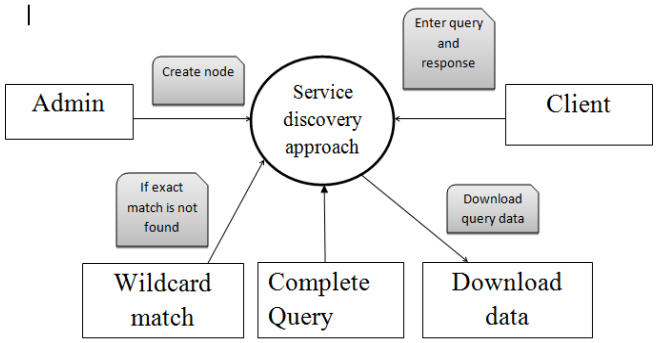


Figure. 2: Data flow diagram for scalable and Reliable Approach for prime information handiness in commission Oriented computing setting

IV. SERVICE DESCRIPTION

Service description supported by the Chord4s contains three parts those are as follows: service identifiers, Specification of QoS and Syntax specification.

A. Service identifier

It contains service symbol for identifying the services out there within the network. 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, 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. Therefore, descriptions of the functionally equivalent services will be distributed to successor nodes adjacent to each other within a certain virtual segment of the identifier circle.

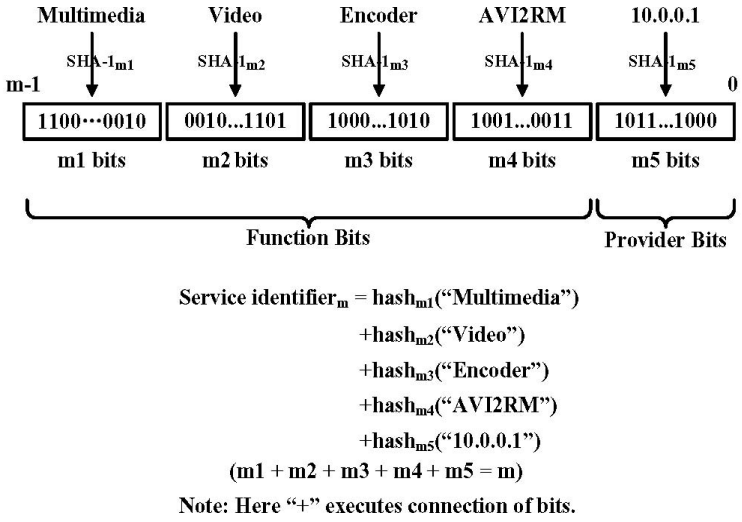


Figure.3. Service identifier generated from hierarchical service descriptions

B. Specification of QoS

QoS awareness is a critical issue in an SOC environment. QoS-aware service discovery should filter out the services that cannot meet service consumer's QoS requirements and only return the ones that can. 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.

V. SERVICE PUBLICATION

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

A. Traditional approach in Chord

In Chord, basic principle is to store the data or the pointer to the data at the first node whose identifier is equal to or follows the identifier of the data in the identifier space. Chord uses SHA-1 as its hashing function to generate identifiers for the data and nodes. The generic primitives used in Chord are as follows:

Identifier node ¼ hashSHA_1ðIPnodeÞ;

Identifier data ¼ hashSHA_1ðDescriptiondataÞ*:*

When distribution or lookup needs to be performed, the following primitives will be used:

Put identifier data; data=pointer to data Þ;

Lookup identifier data Þ:

The put function will store the data or the pointer to the data at the successor node whose identifier is equal to or follows the parameter identifier, while the lookup function will yield the IP address of the node responsible for the required identifier.

B. Service Description Distribution in Chord4S

In this section, how to design Chord4S-based systems and applications to facilitate even service description distribution is discussed. Consider a Chord4S-based network overlay consisting of n nodes, let the length of the service identifier be m and the maximum number of functionally equivalent services be k. The length of the provider bits x should be carefully calculated to achieve even service description distribution. Obviously, a smallest virtual segment should be capable of accommodating all the functionally equivalent services, as constraint (1) below:

$$2^x \geq k-1 \qquad (1)$$

$$x \geq \log_2(k-1) \qquad (2)$$

Hashed into the identifier space, the n nodes are distributed on the Chord4S circle with as the average distance between each other. So to accommodate k functionally equivalent services in a smallest virtual segment, the capability of the virtual segment is supposed to be (K-1). Then to allocate enough bits for provider bits, constraint (3) below should be satisfied

$$2^x \geq (k-1).2^m/n \qquad (3)$$

the descriptions of functionally equivalent services can be evenly distributed in a virtual segment which means that all of them are distributed to different successor nodes.

$$x \geq \log 2(k-1/n \ .2^m) \qquad (4)$$

VI. SERVICE QUESTION

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

A. Service Specific Question

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,

B. Question with wildcard(s)

Sometimes service consumers need to search for categories of services.

C. Pseudocode for finding successor operation

The pseudocode that implements the service discovery process is shown in Fig.4. When looking for matched successor nodes, node n checks the entries of its finger table to find the node with the smallest identifier in the target virtual segment which is about to take responsibility of keeping routing the query.

```
// When node n receives a query message
n.find_successor(message)
if(message.identifier <= identifier)
if message.counter == 0
return message;
end if
for i = 1 upto service.length
if match_making(message.service[i])
message.counter = message.counter – 1;
message.candidate _service_provider.add
(service[i].descriptions);
If message.counter == 0 break
end if
end if
end for
n’ = find_next_successor(message.identifier);
if n’ != null
return n’.find_successor(message);
else return message;
end if
end if
if(message.counter != 0)
n’ = closest_preceding_finger(message.identifier);
return n’.find_successor(message);
end if
end n.find_successor(message)
//return next closest successor node
n.find_next_successor(identifier)
max_identifier = get_max_potential_node_identifier;
min_ identifier = get_min_potential_node_identifier;
for i = 1 upto finger[].length
if(finger[i].node Є (min_identifier,max_identifier))
return finger[i].node;
end if
end for
return n;
end n.find_next_successor(identifier)
//return closest finger preceding identifier
n.closest_preceding_finger(identifier)
for i = finger[].length downto 1
if(finger[i].node Є (n,identifier))
return finger[i].node;
end if
```



```
return n;
end n.closest_preceding_finger(identifier)
```
Figure.4. Pseudocode for finding successor operation

D. Pseudocode for function of match making

If the service meets the QoS requirements, the successor nodes adds the corresponding service description into the list of candidate service providers in the query message and then forward the query message. Otherwise, the query message will be simply forwarded according to the routing protocol.

```
// the match making function
n.match_making(message,services[j])
ifis_matched(message.identifier, services[j].identifier)
// check if a service meets the QoS requirements
for i = 1 upto message.QoS.entries.length
if is_unmatched(message.QoS.entries[i],
services[j].QoS.entries[i])
return false;
end if
end for
return true
end if
return false
end n.match_making(message,services[j])
```
Figure.5. Pseudocode for function of match making

VII. EXPERIMENTAL EVALUATION

Chord4s protocol is using for the service description distribution and discovery. It organizes the nodes in the form of ring. 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.

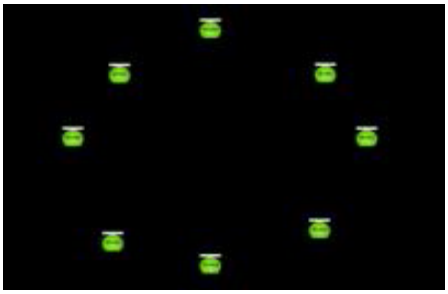
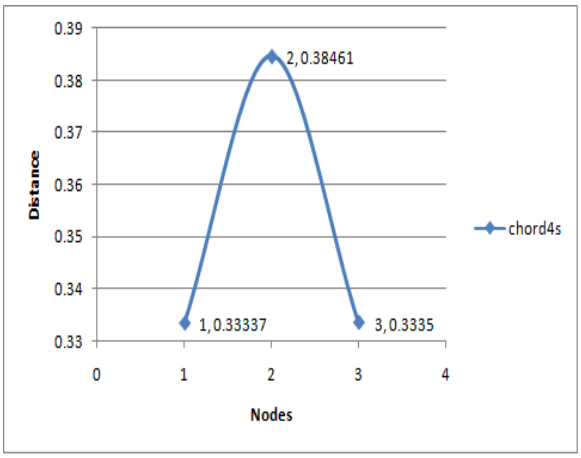


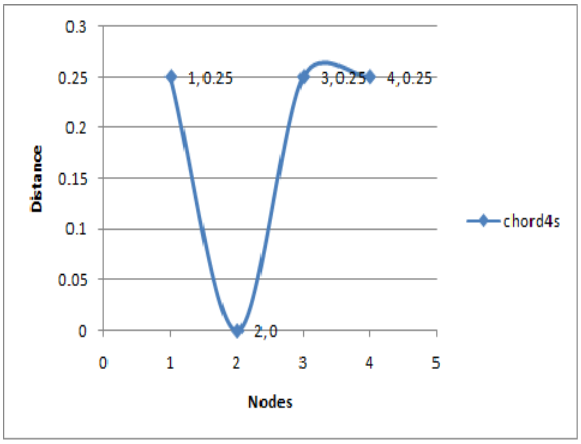
Figure.(a). Network creation

Chord4S is good, 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. In Fig.(a) there are three nodes present. Client request for service after it takes service from specific number of nodes with distance. It means client not get fail. Fig.(b) (c)and (d) shows the graph for 3,4,8 nodes respectively. In Table 1 number of experiments are calculated, in this First Experiment three nodes present and providing the service is successfully so result shows success. After second Experiment four nodes present when client requesting query node providing the service there is no fail in service response. Also in third experiment eight nodes present and they providing the service successfully. All this service selection is done with Wildcard(s).

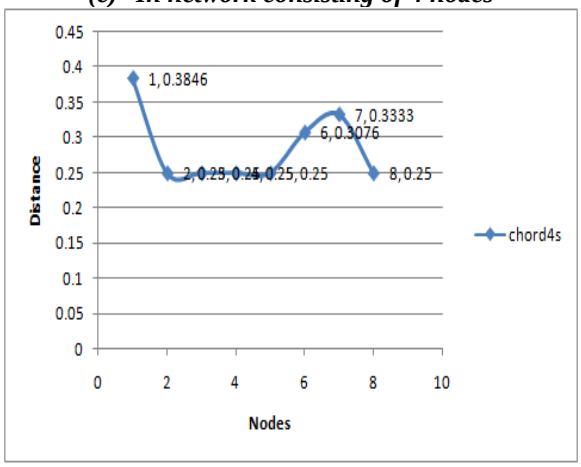
Routing performance for queries with wildcard(s) in that the nodes verses number of services shown in Fig.(e). This routing performance calculate from query with wildcard(s). The service discovery with wildcard query gives good searching results. Experimental description demonstrate that Chord4s achieve high data availability.



(b) In Network consisting of 3 nodes



(c) In network consisting of 4 nodes



(d) In network consisting of 8 nodes

Number of Experiments	Number of Nodes	Number of Services	Result
1	3	3	Success
2	4	4	Success
3	8	8	Success

TABLE 1. : Specifies of Experiments on Service Discovery with Wildcard(s)

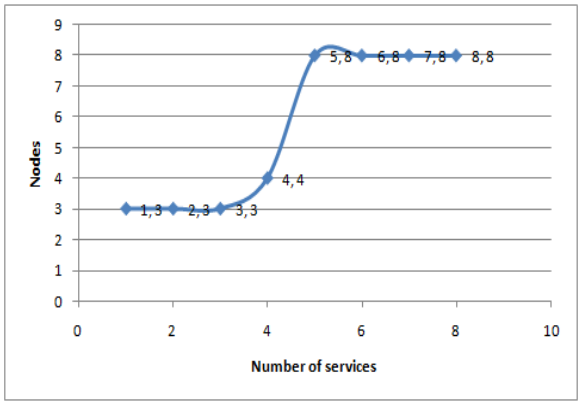


Figure.(e). Routing performance for query with wildcard(s)
Figure.6. Data availability in Chord4S

VIII. 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. An efficient routing algorithm is developed to facilitate queries of multiple functionally equivalent services. Chord4S is scalable, reliable, and robust due to the enhanced peer-to-peer architecture. Experimental results demonstrate that Chord4S can achieve high data availability and efficient query of multiple functionally equivalent services with reasonable overhead. In the future, integration of semantic information of services into Chord4S using popular tools, such as Petri Net and WSMO, will be investigated in order to increase the flexibility and accuracy of the service discovery.

REFRENCES

[1] 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.
[2] 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.
[3] 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.
[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. Conf. SIGCOMM, pp. 161-172, 2001.
[5] 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.
[6] 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.
[7] 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.
[8] S. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to- Peer Internet Telephony Protocol," Proc. IEEE INFOCOM, pp. 1- 11, 2006.
[9] 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.
[10] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tueck, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems," Proc. Eighth Int'l Workshop Job Scheduling Strategies for Parallel Processing (JSSPP '02), 2002.

