Chord4TSD: A Decentralized Trust Based Service Discovery Approach on Peer-to-Peer Networks

Yogini Bhamare

Abstract: ***Service-Oriented Computing (SOC) is emerging as a standard for developing distributed applications and having reliable, scalable 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 being used which can suffer from problems like performance bottleneck and vulnerability to failures. To overcome these problems, this paper proposes a peer-to-peer-based decentralized trust based service discovery approach named Chord4TSD. Existing decentralized service discovery approaches may also suffer from problems like data loss because of node failure and access of malicious services. Proposed Chord4TSD distributes and discovers trusted services in a decentralized manner by utilizing the data distribution and lookup capabilities of Chord4S. Functionally equivalent services are published to different successor nodes, to improve data availability, that are organized into virtual segments in the Chord4TSD. Chord4TSD supports QoS-aware trusted service discovery, based on the service publication approach. Service discovery with wildcard(s) and efficient discovery of multiple services with a single query is also supported in Chord4TSD. It considers trust factor for node and service, before fetching the service from particular node, to avoid accessing malicious services from the network. In addition, the Chord4TSD routing protocol is extended to provide WSDL file to the service consumer, which is generated at the time of service publication based on the service description provided.***

Keywords: Chord4S, DHT, Peer-to-Peer (P2P), SOC

1. Introduction

Service-Oriented Computing (SOC) is emerging as a standard for developing distributed applications, but having reliable, scalable and robust service discovery mechanism is critical issues of utilizing SOC. Traditional service discovery approaches for the web services technology are based on Universal Description, Discovery, and Integration (UDDI) [5]. In traditional service discovery methods for large scalable service networks, centralized registries are being used which can suffer from problems such as performance bottleneck and vulnerability to failures because of a large number of service consumers and requests in an open SOC environment. Due to these disadvantages, applying web services in large scalable environment is mostly prevented. In the distributed SOC environment, to address above issue and to achieve service discovery in scalable, reliable and robust manner, decentralized approach seems to the most efficient way. The Peer-to-Peer (P2P) technology eliminates centralized infrastructures to provide a universal approach for improving scalability, robustness and reliability of distributed systems. In the 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 approach, continuous research is going on in the field of SOC.

In Peer-to-Peer approach, set of distributed nodes are present which forms the P2P network. When a provider registers the new service, it is stored into the repository after assigning it to the relative service node. The consumer can submit service query to any of the nodes of the network and if that node doesn"t contain the required service description then query is routed to the respective node. Description of the matched query is then retrieved from that node and returned to the service consumer as a query result. This is the overall idea about how exactly service request is exactly processed in P2P based decentralized service discovery approach. To implement such a service discovery approach, Chord-based and DHT based approaches are studied by different researchers.

By making use of Distributed Hash Table (DHT), even data distribution to nodes and efficient query routing can be achieved in structured P2P systems. But in DHT based systems, functionally equivalent service descriptions are distributed on the same successor node because hashing value is similar for all these nodes. If such a node fails, then any of these services will not be available to the consumer and hence DHT based P2P approaches to decentralized (P2P based) service discovery may not be that efficient in terms of service availability. This disadvantage can result in serious problems in open and dynamic SOC environments in which unexpected failure of nodes cannot be avoided [8]. In Chord-based approach, Chord is used to facilitate decentralized web service discovery [3]. Emekc¸i et al. [3] present a P2P framework based on Chord for web service discovery which uses finite automata to represent web services. But such approaches are also vulnerable to the issue of data availability in open and volatile SOC environments. Descriptions of multiple functionally equivalent services would be stored at the same successor nodes and it may lead to severe data loss in case of such node failures.

This paper proposes Chord4TSD, a Chord-based decentralized trust based service discovery approach that supports service description, distribution and discovery in a Peer to Peer manner. The main purpose of designing the Chord4TSD is to support trustworthy service discovery as trust is one of the most important factors in decision making by a service consumer, requiring the analysis and evaluation of the trustworthiness [13] of a service provider. It considers trust factor for the node as well as service, before fetching the service from particular node, to avoid accessing malicious service in a network. Chord4TSD largely improves the availability of service descriptions, by distributing descriptions of functionally equivalent services to different successor nodes, in volatile environments. In case if one node fails, then a service consumer can still find functionally equivalent services that are stored at other successor nodes. It supports service discovery with wildcard(s) and QoS awareness. Going forward, Chord4TSD extends Chord‟s original routing protocol for supporting discovery of multiple functionally equivalent services to different successor nodes with a single query which is necessary for selection of optimal service providers [12]. Also it generates and provides corresponding WSDL file to service consumer so that it can be used further as per the requirements.

2. Literature Survey

2.1 Study of Centralized Service Discovery Approaches

The centralized client/server model has been opted for service discovery in SOC [7]. These traditional service discovery approaches of web services technology are based on Universal Description, Discovery, and Integration (UDDI) [5]. As explained in [5], the focus of Universal Description, Discovery & Integration (UDDI) is the definition of a set of services which supports the description and discovery of (1) organizations, businesses and other Web services providers, (2) Web services which they make available, and (3) technical interfaces which may be used to access those services. Extension for a query federation of UDDI [11] registries within a Web Service environment is presented in [10] by Rompothong and Senivongse. But the main limitation here is authors did not provide any experimental evaluation for this. Wu et al. [2] describe an interoperable model of distributed UDDI. The model divides UDDI servers into three types: normal server, super domain server and root server. Here philosophy of Domain Name System (DNS) is adopted. Super domain servers, which are managed by a root server, are used to maintain normal servers. This model is exposed to the same threats that DNS faces, e.g., Distributed Denial of Service (DDoS) attack, as it is based on a concept of DNS.

2.2 Study of Peer-To-Peer-based Decentralized Service Discovery Approaches

Decentralized service discovery approach is considered as a promising approach to address the problems caused by centralized infrastructures. Some preliminary research has been already conducted for utilizing P2P computing for service discovery. In [4], a multicast discovery protocol, the Web Services Dynamic Discovery (WS-Discovery), for locating services on a local network is developed by Canon, Intel, Microsoft, BEP Systems, and WebMethods. In the WS -Discovery protocol, request is sent to the respective multicast group by a client to locate a target service. To scale to a large number of endpoints, protocols define the multicast suppression behavior when a discovery proxy is available on the network and can be switched on. Need for polling is minimized for the target [7] services that wish to be discovered by sending an announcement when they join and leave network. WS-Discovery is becoming popular and is already being used by some software vendors, such as the "People Near Me" contact location system in Microsoft‟s Windows Vista operating system [8]. But WS-Discovery is specific for ad hoc networks and still there is no any successful experience in applying WS-Discovery in large-scale SOC environments. Sapkota et al. [6] propose distributed web service discovery architecture. This is based on distributed shared space concept and intelligent search among a subset of spaces. Web service descriptions‟ publishing as well as for submitting requests to discover the Web service of user‟s interests is allowed. Integration of applications which are running on different [7] resource specific devices is supported as well. But in its current implementation, the shared space is still centralized and no experimental evaluation has been provided to evaluate the proposed architecture [8]. In [9], Hu and Seneviratne propose the approach which is based on a concept that service providers themselves should take responsibility to maintain their own service descriptions in a decentralized environment. For grouping peer nodes by service categories to form the islands on Chord ring, decentralized service directory infrastructure [7] is built with hashing descriptive strings into the identifiers. To handle the routing across islands and within the islands[7], Island Table and Native Table are created on every peer node respectively.

Schmidt and Parashar in [1] describe a system that supports complex queries containing keywords, partial keywords and wildcards by implementing an Internet-scale DHT. The system assures that all the existing data elements matching a query will be returned in terms of count of number of messages and number of nodes involved. To map the multidimensional information space to physical peers effectively, key innovation scheme, a dimension reducing indexing scheme is used. It provides Chord having the ability to perform metric-based similarity search. Node failures would be leading to severe data loss when above approaches in [3] and [1] are adopted to provide service discovery because the descriptions of the functionally equivalent services would be stored at the same successor nodes. Detailed literature survey for this study has been published in [7].

The research reported in this paper is similar to the work presented in [8], i.e. Chord4S, using layered service identifiers to control the distribution of service descriptions. However, this research addresses the important issue of trustworthy service discovery in opened volatile SOC environments. Hence, along with efficient QoS-aware service discovery and service discovery with wildcard(s), below points are also covered in the proposed approach.

- Service discovery as per service consumer defined trust threshold. Trust for nodes and rating for each service is considered
- WSDL file is generated by the system based on service description provided and fetched at the time of service discovery
- Check for specific Boolean attribute „Available‟. When it is set as False, service should not be available in network
- No simulation used

3. Proposed System

This research addresses the issue of trustworthy service discovery in opened volatile SOC environments without losing data availability in open and volatile SOC environments. It supports efficient QoS-aware service discovery and service discovery with wildcard(s).

3.1 Service Description

There are four main parts of service description supported by Chord4TSD, i.e. service identifier, QoS specification, Trust Threshold and syntax specification. This paper only focuses on the first three parts, i.e., service identifier, QoS specification and trust threshold. The syntax specification describes the syntax (names and data types of the input and output parameters) of the service. It is not addressed here as it is usually used during the invocation of a service which is not in scope of this paper.

Service Identifiers: Distribution and query for hierarchical service description is supported by Chord4TSD, e.g., "Booking. Flight. France. AirFrance". The number and the order of the layers are application specific and can be determined by the designers of the applications. Considering this hierarchical service description, a service identifier in Chord4TSD is divided into two parts, i.e. function bits and provider bits. Function bits are used to refer the functionality of the service whereas provider bits are used to describe information about the provider of that particular service. Service identifier is generated by hashing the service description in which Chord4TSDallocates some bits of the service identifier for the function description and remaining for the provider bits. The sample service identifier is shown in Figure. 1. Functional service matchmaking in service discovery is done using function bits. The main purpose of maintaining provider bits is to distinguish the functionally equivalent services and distribute them accordingly. Service descriptions and provider information is hashed using CRC32 to further generate the service identifier. In Chord4TSD, the identifiers are organized in an ascending order in a circle. By this way, 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. Globally, a Chord4TSD circle is composed of a number of virtual segments and each of these contains service identifiers from a group of functionally equivalent services.

Chord4TSD also allows service descriptions in mixed structures. e.g., there is an application with a maximum of five layers, i.e. four layers for functional bits and fifth layer for provider bits, a service description like "Booking. Flight. France" which is 4 layers (3+1) only is also acceptable. To generate a service identifier for this type of service description, service identifier for the first three layers will be generated by using hashing. i.e. hashm1 ("Booking"), hashm2 ("Flight"), and hashm3 ("France"). The fourth layer would be zero by default. In this scenario, the service description will be placed in a virtual segment containing all the service descriptions starting with "Booking. Flight." A simplified Chord4TSD circle is shown in Figure. 2 to present the specific situation.

QoS Specification: Service consumers usually have specific Quality of Specification requirements and hence service discovery approaches should take it into consideration. The services that cannot meet service consumer‟s QoS requirements should be filtered out by QoS-aware service

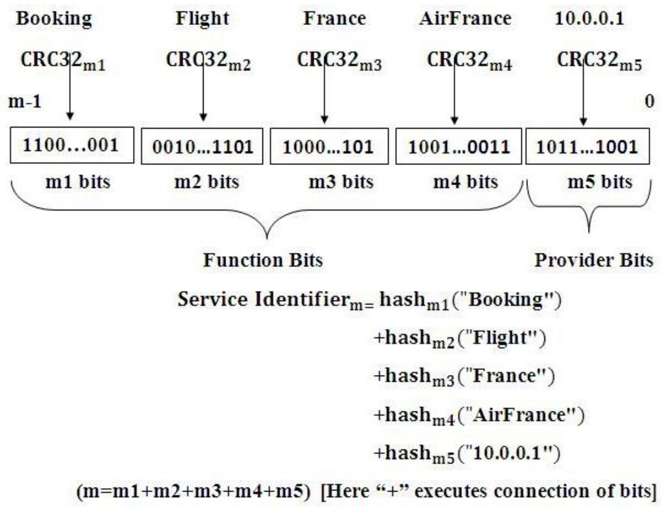


Figure 1: Service identifier generated from hierarchical service description

discovery and only return the ones that can. In Chord4TSD, service providers are allowed to publish their services with quality specifications attached as advertisements. After finding matched the service description of functional requirements, the service consumer can then look over the attached quality specification. Following two types of QoS attributes are supported by Chord4TSD,

- *Numeric QoS attributes:* A QoS attributes that can be assigned with any value selected from a numeric interval is nothing but the numeric QoS attribute. QoS attributes such as price, execution time, availability, etc. fall into this category. Comparison operators, e.g., <;<=;>;>=etc., are used to specify service consumer‟s QoS requirements of this type, such as "Price < =$1:000:00".
- *Boolean QoS attributes:* A QoS attributes that can be assigned to one of the two values: true and false is nothing but the Boolean QoS attribute. For example, a hotel booking service may have a QoS attribute such as Cancellable that can be assigned with either true or false specifying that the booking can or cannot be cancelled. Two comparison operators, ==and! = can be used to specify QoS requirements of this type, such as "Cancellable = = True."Also, there is a specific check provided for „Available" attribute. When it is set to False, that service will not be available in network, Service can only be consumed, when it has „Available" attribute value as True.

Individual QoS requirements can be combined using logical connectives, when a service consumer has requirements of multiple QoS attributes.

Trust Threshold: Service consumers always want to fetch services from trusted nodes to get rid of malicious services available in the network. The services that are having trust factor below service consumers required trust threshold should be filtered out in this trustworthy service discovery.

To achieve this, trust factor for each node (trust of Other Connected Nodes to Trust on current node) is set when network is created. Initial node trust is set to 1 for all the nodes in the network. Later when service is accessed from any perticular node, that node‟s trust value is increased by 1. This value is increased upto 10 and once this value reaches to 10, it is kept as 10 for that node. Along with the node trust, service trust is also considered here. While publishing any service in a network, 3 nearest nodes are selected from the network and rating for the service to be published is taken from those selected nodes and set for that service. At the time of service discovery, trust for a service is calculated by considering data of trust for service and corresponding nodes (from 3 different nodes). This calculation is done as given below:

Let‟s consider,

- p1;p2;...pi;...;pl;...;pn be the nodes in a network.
- S be the service available on nodes from different providers.
- Tp1p2 is the trust of node p1 on node p2; Tp1p3 is trust of node p1 on node p3; and so on….
- The rating of node p2 about service S is Rp2S; rating of node p3 about service S is Rp3S; and so on…

Trust for a Service S for node p1 is calculated as,

$$T_{p1S} = \left(\frac{T_{p1p2} \cdot R_{p2S} + T_{p1p3} \cdot R_{p3S} + T_{p1p4} \cdot R_{p4S}}{T_{p1p2} + T_{p1p3} + T_{p1p4}} \right)$$

At the time of service discovery, after finding matched service description of functional requirements and attached quality specification, search is processed further with respect to trust threshold defined by service consumer. If calculated trust is greater than defined threshold, and then it will be countable other wise move forward to discover other service which service follows the constraint.

3.2 Service Publication

Data availability is improved in Chord4TSDby distributing descriptions of functionally equivalent services to different nodes. By this way, a failed node would just have limited impact on data availability and a service consumer has chance to locate the functionally equivalent services from other available nodes.

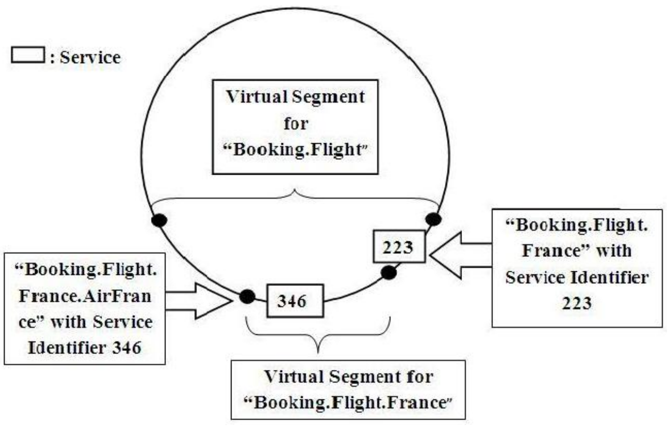

: Service
Virtual Segment for "Booking.Flight"
"Booking.Flight.France.AirFrance" with Service Identifier 346
346
223
"Booking.Flight.France" with Service Identifier 223
Virtual Segment for "Booking.Flight.France"

Figure 2: Virtual segments for "Booking. Flight" and "Booking.Flight.France."

To guarantee such data availability of Chord4TSD based systems, few design specification needs to be taken into consideration like Chord4S [8]. Consider, N = network consisting of N nodes, l = length of service identifier, P = maximum number of functionally equivalent services and x = length of provider bits and this x should be carefully calculated to achieve even service description distribution. A smallest virtual segment should be capable of accommodating all the functionally equivalent services, as constraint (1) given below.

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

Hence,

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

If n nodes are distributed on the Chord4TSD circle then they are distributed with $\frac{2^l}{N}$ as the average distance between each other. Hence in a smallest virtual segment, to accommodate P functionally equivalent services, the capability of the virtual segment is supposed to be $(P-1).\frac{2^l}{N}$. Then to allocate enough bits for provider bits, constraint (3) given below should be satisfied.

$$2^x \geq (P - 1).\frac{2^l}{N} \qquad (3)$$

Hence,

$$x \geq \log_2 \left(\frac{P-1}{N}.2^l\right) \qquad (4)$$

With constraints (2) and (4) satisfied, 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.

Also, WSDL file is generated from Service name and service descriptions and stored along with other service details in Chord4TSD. Service name provided while registering the service is taken as Service and descriptions are taken as methods in WSDL file. At the time of Service discovery, WSDL file will also be fetched along with the other service details so that service consumer can use/enhance it further as per the requirements.

Following strategy is followed while implementing above described service publication approach:

a) To assign provider bits, total number of nodes in network is considered. If number of nodes in network are N, then the length for provide bits is set as,
   - $N \leq 2$^2=> Provider Length = 2 digit,
   - $N \leq 2$^3 => Provider Length = 3digit and so
b) To Publish the Service, when similar service is not already exists in a network, calculate publisher node as per the details given in Table 1.
c) When another equivalent service is created next time, comparisons of service identifiers are done.
   - If Service Identifier for new service is greater than existing service, then new service would be published on successor node
   - If Service Identifier for new service is less than existing service, then new service would be published on predecessor node
   - If it is between two service identifier then service is published to previous highest service identifier node

Table 1: Publisher Node calculation

Layers of Description	1	2	3	4
NodeId > ((N/2) + 1)	(N/2) + (N/4)- 2	(N/2) + (N/4) - 1	(N/2) + (N/4)	(N/2) + (N/4) + 1
NodeId > ((N/4) + 1)	(N/2) - 1	(N/2)	(N/2) + 1	(N/2) + 2
NodeId<= ((N/4) + 1)	(N/2) - 2	(N/2) - 1	(N/2)	(N/2) + 1

d) To restrict virtual segment length, minimum and maximum boundary is defined by using following constraints:

- $Minimum = \frac{N}{4}$
- $Maximum = \left(\frac{N}{2}\right) + \left(\frac{N}{4}\right) + 1$

3.3 Service Discovery

This section gives details of how routing of query messages is performed in Chord4TSD based on the service publication approach described in the above section.

Chord4TSD supports 2 types of queries, i.e. Service specific queries and wildcard queries. Complete details of a service description are present in a service-specific query and it is used to look up a specific service. In a system which allows four-layered function bits in the service descriptions, is a typical example of service-specific query is “Booking. Flight. France. AirFrance”. To compose the query with explicit service information, the service consumer needs to fill in all the layers to initiate a service-specific query. Then each of those layers will be hashed and the results will be connected to generate the function bits of the target service identifier. The provider bits of the query will be stuffed with 0s as the objective here is to look up a group of functionally equivalent services provided by different service providers.

Sometimes service consumers need to search for categories of services. In such cases, service queries using wildcard(s) are necessary, e.g., “Booking.India.Flight.*”, “Booking. India.Raliway.*”, and “Booking.India.*.*”. To solve a query with wildcard(s), it actually looks up a virtual segment composed by nodes succeeding service descriptions which fall into the target service category. The generation of the target service identifier for a query with wildcard(s) is similar to that for a service-specific query. The difference here is that the layers corresponding to the wildcard(s) in query with wildcard(s) will be stuffed with 0s.

Forwarding Service-Specific Queries:To find multiple functionally equivalent services with one query in Chord4TSD, the query must be routed across the corresponding virtual segment of the identifier circle until sufficient services required by the service consumer have been found. In this research, Chord4TSDsupports further routing of a query to other nodes when it reaches a matched successor node.

Each initiated query message contains the following basic information: a target service identifier and a counter. Target service identifier includes function bits and provider bits with the provider bits stuffed with 0‟s. For finding out if a service matching succeeds, binary AND operation is performed by a node between each of its succeeding service identifiers and the target service identifier in the query. The matching succeeds, if the result equals to the target service identifier. This AND operation is used, Logically, to extract the identifier of the virtual segment that the node belongs to and to find out if this identifier equals to the function bits in the required service identifier. Figure. 3.shows a sample of matched service. As the result of the AND operation is equal to the target service identifier, the service description which is stored at that node meet the service consumer‟s requirements. After finding out a matched service description, a query will still be passed along the circle until sufficient service providers are found.

Service consumers include their prespecified QoS requirements and trust threshold (between the scales of 1 to 10) in the query messages, to request trustworthy services with QoS constraints. The process of looking up services is then divided into two steps: functional and nonfunctional. At the 1st step, the query message is forwarded by nodes following the routing protocol. Once the query message reaches a successor node that stores a matched service description, the service discovery proceeds to the second step where successor node checks the entries of the QoS requirements one by one to see if the service meets the QoS requirements specified in the query message. If the service meets the QoS requirements, then it proceeds with the checking for required trust threshold. If the criteria for trust threshold fulfilled, then the corresponding service description is first added (by successor node) into the list of candidate service providers in the query message and then forward the query message (or returns the query message to the service consumer if the discovery process completes). Else, the query message will be simply forwarded according to the routing protocol, when multiple occurrences of services need to be fetched using single query. In this scenario of getting multiple occurrences of same service, at matched node, after meeting QoS requirements and trust threshold that encountered matched successor nodes must perform below three tasks:

- Get *min (query. Counter, m)* matched the service description it contains with m being the number of matched service descriptions, and add it into a query message
- Subtract the value of the counter by *min (query. Counter, m)*
- Check if counter equals to 0. If yes, send a query message back to service consumer, else route the query to a next successor node as per the defined protocol.

Forwarding Queries with Wildcard(s): The process of forwarding queries with wildcard(s) issimilar to that of forwarding service-specific queries, only the bits generated from explicit service information will be taken into consideration, instead of all the function bits. During the process of forwarding a query with wildcard(s), because this type of queries is used to look up services that belong to a specified service category, all the successornodes storing service descriptions that fall into the specifiedservice categories are considered matched. As the result from the AND operation equals to the target service category identifier, the service descriptions which are stored at the node fall into the service category required by the service consumer. To serve for a service query with wild card(s), since a service category corresponds to a virtual segmenting the Chord4TSD circle, corresponding virtual segment needs to be traversed.

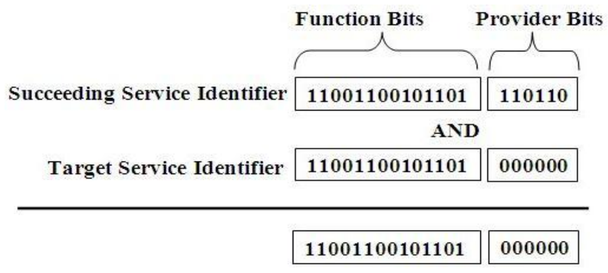


Figure 3: Service matching operation for forwarding a service-specific query

4. Results

On the basis of trust threshold some results are generated by comparing Chord4S with trust based Chord4S (i.e. Chord4TSD) protocol. This protocol has been implemented in java 1.6 on windows OS. Wired/wireless connection is created among the nodes to form a ring topology of a network. Request Eligibility verification, Confidence of Genuinity for node, Time taken for service discovery were evaluated particularly because they are of great importance in Chord4TSD, i.e. trust based service discovery. This experimental evaluation was performed in network consisting of 8 nodes with 5 different types of service requests in order to evaluate the performance of Chord4TSD.

4.1 Request Eligible Services

To evaluate request eligibility criteria, 5 different types of requests have been evaluated. After matching the service description details, QoS parameters are matched. In proposed work Chord4TSD, trust threshold is also needs to be checked. As the value of trust threshold increases, request eligible nodes will be comparatively low because of filter of trust threshold. Hence, less no of services would be returned as a result of service discovery. Reason is that requirement of high trust threshold increase the level of difficulty to find satisfactory service descriptions, and hence results in less eligible services. Figure 4. demonstrates result of request eligible services. Less number of services discovered in Chord4TSD, to avail trustworthy services, is acceptable.

4.2 Confidence of Genuinity

A unique feature, and also a main design goal of Chord4TSD is the trusted service discovery from a network. The result shown in Figure 5. compares confidence a node (requesting node) can have on set of nodes and their services on which the requested service is discovered. The confidence statistics here not only checks for genuinity of node alone but also of service it hosts. This statistics is used to calculate how much requesting node can be confident about the service which is discovered to access it in the future. E.g. suppose there are 3 nodes which host the requested services and are shortlisted by applying the trust threshold criteria.

Now the requesting node calculates the confidence of genuinity of a service and node on the basis of following formula:

Confidence (%) = Average (Service Rating) * Average (Node Trust) * 100%
Where,
Average (Service Rating) = Sum of Rating of Discovered Services/ Number of Discovered Services
Node Trust= Sum of Trust of Nodes/ Number of Nodes (on which service is discovered)

Result after the service was successfully discovered and after applying the threshold trust criteria is shown in Table 2.

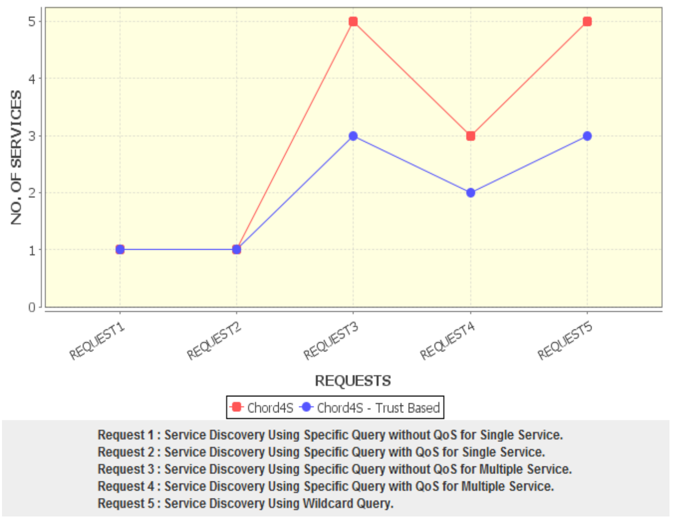
Table 2: Result of Average Rating and Node Trust

Node Shortlisted	*Service*	*Service Rating*	*Node Trust*
Node1	booking	8	8
Node2	booking	6	10
Node3	booking	7	9
Total		21	27
Average		7	9

Confidence (%) = 7 * 9 * 100% = 63%

Hence the requesting node has at least 63% confidence while accessing any of the services from any of the nodes discovered of its genuinity.

For Chord4S, confidence of genuinity is calculated as,
- Value1= Confidence [Chor4TSD] (%) – 10
- Take random value in the rage from 10 to Value1

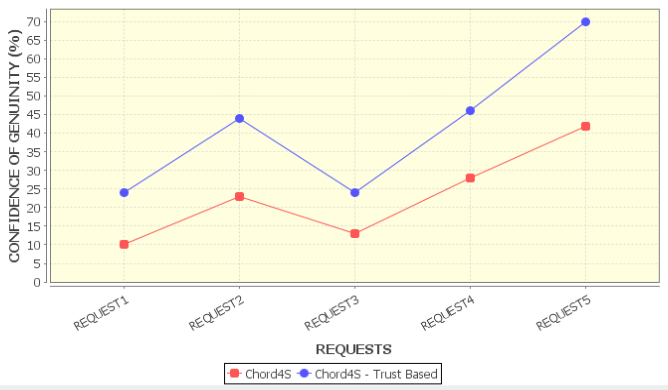


Request 1 : Service Discovery Using Specific Query without QoS for Single Service.
Request 2 : Service Discovery Using Specific Query with QoS for Single Service.
Request 3 : Service Discovery Using Specific Query without QoS for Multiple Service.
Request 4 : Service Discovery Using Specific Query with QoS for Multiple Service.
Request 5 : Service Discovery Using Wildcard Query.

Figure 4: Request Eligible Services

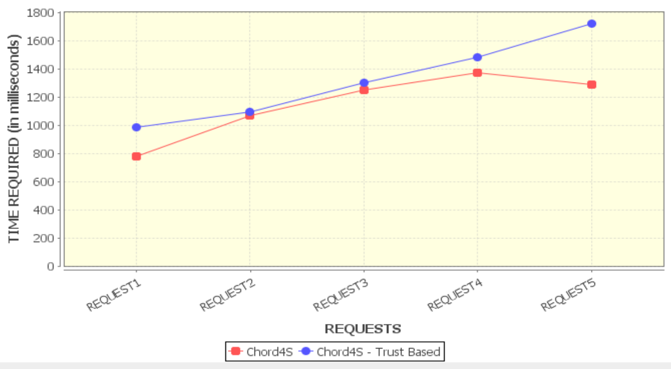
4.3 Time required for service discovery

To analyze time requirement for service discovery, 5 different types of requests have been evaluated. In both the systems, after matching service discovery, QoS requirements are matched. As the number of QoS attributes that the service consumers have requirements are increases, the average number of hops needed and hence overall time requirement, to complete routing the query messages will increase accordingly. The reason is that requirements for more QoS attributes increase the level of difficulty to find satisfactory service descriptions, and hence requires visiting more successor nodes which results in more time requirement. In addition, in Chord4TSD, some more time is required for the calculations and comparisons done for trustworthy service discovery. Figure. 6. demonstrates result of time requirement for service discovery. Slightly more time is required in Chord4TSD and it is acceptable to get rid of access to malicious services and discover only trustworthy services.



Request 1 : Service Discovery Using Specific Query without QoS for Single Service.
Request 2 : Service Discovery Using Specific Query with QoS for Single Service.
Request 3 : Service Discovery Using Specific Query without QoS for Multiple Service.
Request 4 : Service Discovery Using Specific Query with QoS for Multiple Service.
Request 5 : Service Discovery Using Wildcard Query.

Figure 5: Confidence of Genuinity



Request 1 : Service Discovery Using Specific Query without QoS for Single Service.
Request 2 : Service Discovery Using Specific Query with QoS for Single Service.
Request 3 : Service Discovery Using Specific Query without QoS for Multiple Service.
Request 4 : Service Discovery Using Specific Query with QoS for Multiple Service.
Request 5 : Service Discovery Using Wildcard Query.

Figure 6: Time required for service discovery

5. Conclusion

Peer-to-peer-based service discovery becomes more efficient and effective after the deficiencies of centralized service discovery are identified. Proposed system Chord4TSD, inherited from Chord4S, using layered service identifiers to control the distribution of service descriptions and achieve high data availability. It supports QoS-aware, trustworthy service discovery and service discovery with wildcard(s). In addition, this routing protocol supports efficient discovery of multiple services with a single query. Also it generates and provides corresponding WSDL file to service consumer so that it can be used further as per the requirements.

Integration of semantic information of services into Chord4TSD will also be investigated and incorporated in WSDL file, in the future, in order to increase the flexibility and accuracy of the service discovery.

References

[1] 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.C. Sch

[2] 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.

[3] F. Emekc¸i, O.D. Sahin, D. Agrawal, and A.E. Abbadi, "A Peer-to-Peer Framework for Web Service Discovery with

[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] 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.

[6] P. Rompothong and T. Senivongse, "A Query Federation of UDDI Registries," Proc. First Int'l Symp. Information and Comm. Technologies, pp. 561-566, 2003

[7] Prof. D.N. Rewadkar, Yogini Bhamare "Different Approaches for Peer-to-Peer Based Decentralized Service Discovery," International Journal of Advanced Research in Computer Science and Software Engineering 3(11), Volume 3, Issue 11, pp. 24-28, 2013.

[8] Qiang He, Member, IEEE, Jun Yan, Yun Yang, RyszardKowalczyk, 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] 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.

[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] 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.

[12] 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.

[13] Wang, Yan, Orgun, Mehmet A., Lim, Ee-Peng, Liu, Guanfeng, “Finding the Optimal Social Trust Path for the Selection of Trustworthy Service Providers in Complex Social Networks,” Services Computing, IEEE Transactions on (Volume:6 , Issue: 2), April-June 2013.

Author Profile

Yogini Bhamare received the B.E(Information Technology) and M.E(Computer Engineering) degrees from Pune University in 2005 and 2014, respectively. She is now with Persistent Systems Ltd
.