2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

Context-aware Service Discovery and Selection in Decentralized Environments

Mariwan Ahmed*, Lu Liu*, Bo Yuan*†, Marcello Trovati* and James Hardy*
*School of Computing and Mathematics, University of Derby, Derby, DE22 1GB, United Kingdom
†The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China
Email: {M.Ahmed, L.Liu, B.Yuan, J.hardy}@derby.ac.uk

***Abstract*—With the increasing use of web services in everyday tasks, we are entering an era of Internet of Services (IoS). Service discovery and selection has become critical issue in the area of web services. Traditional service discovery approaches are often supported by centralized registries that could suffer from single point failure, performance bottleneck, and scalability issues in large scale systems. To address these issues, in this paper we propose a context-aware service discovery and selection approach in a decentralized peer-to-peer environment. In the approach homophily similarity was used for bootstrapping and distribution of nodes. The discovery process is based on the similarity of nodes, previous interaction and behaviour of the nodes, which will help the discovery process in a dynamic environment. Our approach is not only considering service discovery, but also the selection of the best available web service by taking into account the QoS properties of the web services. Experimental results were based on real world semantic web service dataset, the results showed that the approach achieved better performance and efficiency in both discovery and selection processes.**

***Keywords*—decentralized environment; context-aware service discovery; homophily; Web service selection; Quality of Service (QoS); Partial matching**

I. Introduction

In the recent years distributed technologies are introduced through Service Oriented Architecture (SOA), Grid computing and Cloud computing, all of these technologies have been used throughout organizations and companies of different sizes. With the advances of web service technology the web is moving from data- oriented web to service oriented web. Continually the number of web services increases, for example current cloud computing trend which is a Service-Oriented application has been growing in number of web services such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The increasing number of web services has lead to critical issues like efficiency and scalability, however most of the current approaches focus on centralized architectures, in which a single service registry or multiple service registries synchronize together. These approaches lead to bottlenecks, single points of failure, and scalability limitations. In addition, one of the main drawbacks of centralized approaches is having a global knowledge about the system which is not presented in decentralized environments. While availability and demand for services grows significantly, the inherited disadvantages in centralized environments prevent web services from being applied in large scale service networks. As Service Oriented Computing environments are largely distributed, a decentralized approach appears to be a suitable way to address the issues and achieve scalable, reliable and robust service discovery and selection.

Peer-to-Peer (P2P) has been used worldwide for resource sharing and received great success such as in (Gnutella, Chord, Freenet, JXTA and etc.). P2P technology is one of the areas which provides a universal approach in improving reliability, scalability, and robustness of distributed systems by weakening centralized control. In a P2P environment each node can play the role of provider or consumer depending on the need at any moment of time. P2P is located in an overlay network and distributed without any centralized control. There are approaches which use a decentralized system without central supervision to discovery services as in [1–3] . The challenge is to find the service with fewer steps, which considers only local and neighbour information. In this approach we are designing a novel self-organized P2P architecture and social enhanced model for collaborated, dynamic and context-aware service discovery and selection to improve the scalability and efficiency of traditional methods. In this approach homophily was used, meaning that the most noticeable properties of services which determine similarity between nodes in the system is utilized as a method of bootstrapping and creating links between nodes. The discovery process is based on the similarity, previous interaction and social behaviour of nodes, which will help the discovery process and create a more dynamic environment for joining and leaving of the nodes. It is not only about discovering services, it also considers the selection of the best available service by taking into account the QoS properties of the services.

A. Motivation

Web services have become a de facto in the field of computing, and seemingly enter every aspects of life. SOA, Grid, and cloud computing are different paradigms of computing based on web services. With the increase in the number of web services, the requirements of consumers increase seamlessly. A centralised approach cannot cover all aspects of practically unlimited consumer requirements, because it is not feasible for any service repository or service provider to meet all consumer demands. On the other hand, a centralized approach is always prone to single point failures and bottlenecks, which hugely affect the availability and reliability of web services. Therefore, a decentralized P2P approach is proposed where each peer node can represent a cloud platform and provide different services. A practically unlimited number of services can be found outside each cloud platform and can be outsourced based

978-1-5090-0154-5/15 $31.00 © 2015 IEEE
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.329

on the consumer request. The nodes can dynamically join and leave the network which provides an elastic environment for limitless number of services; as a result each node can outsource services from other nodes based on the context and social behaviour. By this way consumer demands will be fulfilled, and the issue of single point failure will be solved using different number of nodes each of them act as a provider and requester for services.

B. Contributions

The paper presents a service discovery and selection algorithm in a decentralized environment which will improve the efficiency and performance of the discovery and selection process in comparison to the previous works in this area. Contributions can be summarized as

- Designing a service discovery algorithm in a decentralized P2P environment considering homophily for node distribution and using social behaviour and node interaction to discover the most relevant services.
- Proposing a service selection algorithm to consider partial matching of the services based on QoS parameters, and a related ranking algorithm capable of ranking all the discovered services and returning the best available service to the consumer.
- Use of real world semantic web service dataset to provide experimental proof of the performance and efficiency improvement of the proposed algorithm when compared to the existing algorithms.
- Implementation of a prototype platform for Peer-to-Peer service discovery and selection in a real world environment.

The rest of the paper is structured as follows: Section II shows and analyses some related work in the field of service discovery and selection. Sections III illustrate system structure and provides detail of the homophily similarity algorithm, and provides details on community creation in decentralized environment and also discuss service discovery and selection. Section IV and V consider network simulation and evaluation. Final section VI presents conclusion on the approach.

II. Related Work

Currently there is a plethora of web services available which have been created by industries and companies and new services continuously emerge. This increase leads to diversity in the type and quality of web services and also diversity in the requirements of the consumer. There are mainly two approaches for service discovery and selection which are either centralized or decentralized.

With centralized approaches, the main entity has global knowledge about services and resources in the system; this entity provides coordination for service management and discovery. As in [4–7] , centralized approaches are suitable for systems with fewer number of web services and limited number of consumer requests, since it is always prone to bottleneck delays and single point failures. In the paper of [8], keyword based search has been used for service discovery, in which provider receives a keyword request for a service, then a list of discovered services will be returned to consumer to select, but accuracy of keyword search is not the best since it does not consider semantics of service. However there are approaches use semantic web to improve accuracy of service discovery and selection as in Agarwal and Studer [9], Plebani and Pernici [10] and Klush, Fries and Sycara [11].

Decentralized approaches can address the issues which existed in the centralized approaches. With a decentralized approach, all the nodes are considered equal without any centralized control; each node can act as a requester and provider. Other approaches use P2P for service discovery and selection in decentralized environment such as super node in [12]. With super nodes it is always possible to have failure in the nodes, and replacing them with less qualified nodes results in a decrease in the overall performance of the system. The Distributed Hash Table (DHT) [13–15] has been proposed in a P2P environment which uses a key identifier to a document or a service. Different approaches for DHT like Chord [12], Pastry [17] and CAN [18] have been proposed. The differences between the approaches are mainly on the distribution of key and the mechanism of joining/leaving of the nodes. The cost of search process in DHT is high and maintenance of the tables is time consuming especially during joining and leaving of nodes in the system.

In [2] a decentralised service discovery approach was presented by considering local information in the service discovery process and used homophily between nodes to establish the links. Our work is similar to that approach, but in this approach homophily similarity was used for bootstrapping and establishing community in the system. The discovery process is not only based on homophily similarity but also considers previous interaction and social behaviour of the nodes, which is going to create a more dynamic environment for joining and leaving of the nodes. Beside the discovery process it also considers the selection of the best available service by taking into account the QoS properties of the services using partial service selection algorithm.

III. Decentralized system structure

The system consists of a set of nodes which are distributed in a decentralised manner; each node in the system does not have a global knowledge about other nodes except direct neighbours. This section explains the structure of the decentralized system, nodes, and the services as contents of the nodes.

Definition 1. The decentralized system can be formalized as $C = \{N, L\}$, such that $N = \{n_1, n_2, n_3, \cdots \quad n_m\}$, where *N* represents the list of nodes in the network, n_i represents the i^{th} node in the system and *m* is the total number of nodes. $L = \{n_i, n_j\}$, $(1 \leq i \leq m \quad and \quad 1 \leq j \leq m \quad where \quad i \neq j)$. *L* is a set of links between different nodes in the system, both *i* and *j* representing two different nodes.

Definition 2. Each node n_i in the system *N* can be formalized as a tuple of $\{S_i, Ne_i, K_i, Q_i, \mu_i\}$ where:

- S_i is the list of services in the i^{th} node
- Ne_i is the list of direct neighbours for i^{th} node, and ε is similarity threshold to determine if neighbour node content is similar to the i^{th} node content.
- K_i is the knowledge index for the i^{th} node, which contains the service categories of neighbouring nodes.

- Q_i is a received query at any given time in the i^{th} node.
- μ_i is the selection function for i^{th} node to determine nodes which a query can be forwarded.

The network is decentralized, each node at least connected to one neighbour node based on the content similarity, and the environment is dynamic in which nodes can join and leave the network. The distribution of queries is not based on flooding or random distribution; it is based on the similarity of the query to the neighbour node which is determined by the functionμ_i.

Definition 3. Service Distribution and node organization Let $S_i = \{s_1, s_2, ...s_n\}$ be list of services for i^{th} node, for each service there is a service tuple: Service tuple, $s_n = \{I_n, O_n, Cat_n, QoS_n\}$ Where, I_n is the list of service inputs; O_n is the list of service outputs; Cat_n represents the user defined service category; QoS_n is the list of quality of service properties for the service *n*. In the context of services, in every node in the network there is a list of services with different functionality and QoS properties. During the discovery process, based on the service request, each node checks its local services for a service which matches the service request. If there is any service matching the request it will return that service to the requester and forward the request to another node which has similarity with the service request. This process continues until the Time To Live (TTL), which is the maximum number of times a query can be forwarded, becomes zero.

A. Homophily in decentralized environment

Homophily was first used by Lazarsfeld and Merton [19] to define interactions and links between two individuals based on their similarities in set of social entities such as ethnicity, religion, or gender to establish links between them. Using same approach homophily became one of the most prominent properties present in complex networks [16]. In this paper homophily is used to determine the similarity between nodes in the decentralized system, which is based on the similarity of service values for Inputs, Outputs and Categories.

To determine the homophily similarity between nodes Cosine Similarity is used. Cosine Similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. To find the relationship between two services, each service is constructed as a vector in the space of node. The cosine between these vectors gives a measure of similarity. Such functions have largely been used in the web space to identify similarity of documents and web pages. It has been one of the most preferred techniques in information retrieval, clustering and has also been applied to pattern recognition and medical diagnosis [20].

The following explains the details of Cosine homophily process. Given two vectors of attributes, S_k and $S{\cdot}_l$, the cosine similarity, cos(θ), can be determined:

$$Hcos(S_k, S{\cdot}_l) = cos(\theta) = \frac{S_k \bullet S{\cdot}_l}{\|S_k\| \|S{\cdot}_l\|}$$

$$Hcos(S_k, S{\cdot}_l) = \frac{\sum_{i=1}^{n} S_k * S{\cdot}_l}{\sqrt{\sum_{i=1}^{n}(S_k)^2} * \sqrt{\sum_{i=1}^{n}(S{\cdot}_l)^2}} \quad (1)$$

Where, vector S_k represents all attributes of S_k $\{s_k in, s_k out, s_k cat\}$, while vector $S{\cdot}_l$ represents all attributes of $S{\cdot}_l$ $\{s{\cdot}_l in, s{\cdot}_l out, s{\cdot}_l cat\}$ as in fig. 1. The result $Hcos(S_k, S{\cdot}_l)$determines the similarity of two service vectors S_k and $S{\cdot}_l$ based on the similarity of their attributes.

$$\begin{array}{c} S_k \\ S{\cdot}_l \end{array} \begin{Bmatrix} s_k in & s_k out & s_k cat \\ s{\cdot}_l in & s{\cdot}_l out & s{\cdot}_l cat \end{Bmatrix}$$

Fig. 1. List of inputs, outputs, and categories for two vectors in Cosine similarity

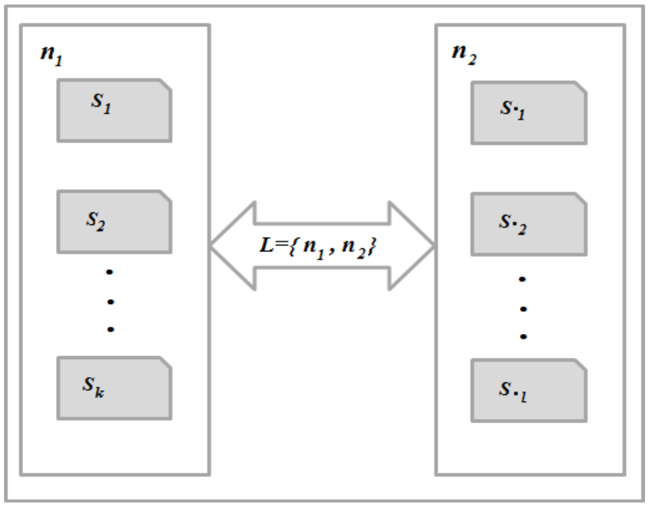

Fig. 2. Relationship between two nodes n_1 and n_2

By using *equation (1)*, similarity between services of both node n_1 andn_2 will be calculated as shown in fig. 2,

$$Hcos(n_1, n_2) = \frac{\sum_{k=1}^{q} \sum_{l=1}^{p} Hcos(S_k, S{\cdot}_l)}{p*q} \quad (2)$$

Where $Hcos(n_1, n_2)$ represents the similarity between two nodes n_1 and n_2 and the value is normalized to be between $[0, 1]$. The value of *q* represents number of services in n_1, while the value of *p* represents number of services in n_2. $Hcos(S_k, S{\cdot}_l)$ represents the similarity between service vector S_k in n_1 and service vector $S{\cdot}_l$ in n_2; *k* represents k^{th} service in n_1 and *l* represents the l^{th} service in n_2 .

To establish the connection between two nodes there is a similarity threshold, ε, the threshold can be set to determine if the similarity of two nodes is in the level to establish connection.

B. Community creation

Once Cosine homophily has been defined in the context of decentralized systems, and then it is described how it is included in both structure creation and service discovery process. Cosine homophily establishes a measure of similarity between two nodes based on services similarity inside the nodes. The similarity measurement is taken into account by nodes during community creation process.

- Each node that is part of the system is considered an entry point. At the beginning each node n_i in the system randomly connects to a number of other nodes in the system.

- The number of random neighbour is determined by the system. Each node n_i should know at least one node n_j that already present in the system.
- The establishment of connection between both n_i and n_j is based on the Cosine homophily between them.

The Probability T_l of establishing a connection between node n_i and node n_j is

$$T_l(n_i, n_j) = \left(\frac{1-Hcos(n_i,n_j)}{T}\right)^{-r} \tag{3}$$

To obtain the probability distribution, the cosine homophily between two nodes should be divided by an appropriate constant *T* that indicates the degree of precision to consider two nodes equal. The *r* parameter is a homophily regulator. When *r* is zero, the system shows no homophily (i.e, nodes are not grouped by similar services). As *r* grows, links tend to connect nodes with more similar services. Basically, *r* makes the system create communities with similar services.

Nodes have a greater chance of establishing connections with other nodes if they provide similar services in the system. As a result of this behaviour, communities of similar nodes are created in a decentralized way. The resulting system structure is a network based on homophily, which grows according to a simple self-organized process. The construction process of a growing network ensures that the oldest nodes have a higher probability of receiving new links than the newest ones.

C. Service discovery

In decentralized structure service discovery relies on the local information in the nodes, which is information about the content of the node and knowledge about neighbouring nodes. In this way it prevents central supervision, eliminates single points of failure in the system, and precludes the need to collect global knowledge about the structure of network which is not available in a large dynamic environment.

In the following section the process of service discovery is explained in a decentralized environment using local information of the nodes and knowledge about first neighbours of the nodes.

During service discovery, when a node n_t sends a request to node n_i, node n_i searches its local services to find any service similar to the request r_t by node n_t. If there is any service similar to the request, it will be returned to n_t. The request will then be forwarded to most promising neighbours of n_i. The most promising neighbours are neighbours with highest score of similarity to the target node n_t. This process of forwarding query continues until TTL becomes zero.

The equation which calculates the most promising neighbour from node n_i is as follows,

$$\mu_i(n_t) = argmax \quad Ps\left(\langle n_j, n_t\rangle\right) \tag{4}$$

For each neighbour of n_j, *Ps* determines the probability that node n_j is a promising neighbour to forward the query

$$Ps(\langle n_j, n_t\rangle) = 1 - \left[1 - \left[\frac{Hcos(n_j,n_t)}{\sum_{n_j \in N_i} Hcos(n_j,n_t)}\right]\right]^{|N_j|} \tag{5}$$

For this probability Cosine homophily-based factors and degree based factors (number of neighbours $|N_j|$) is used to explore the network. For cosine homophily in the discovery process, we use information available in knowledge index of each node about neighbours. In the knowledge index there are categories of services provided by neighbours. These categories are user defined categories, and it is possible to have multiple categories for a single service in a node.

D. Service selection

There are two criteria which service selection is depending on; the first one is QoS from the service tuple, which determines the quality properties of each service. The second one is node Knowledge, which is based on the previous interaction of the nodes and consumer rating for the services in the node.

The selection process considers partial selection algorithm from our previous work [4], which returns the best available web service to the consumer when the providers offer does not fully match every consumer QoS requirements. By having QoS properties from services and node knowledge on one side, and consumer specified QoS requirements on the other side, the normalized value of each QoS property can be calculated by using the case dependant equation (6) (11).

For values with high tendency

$$\frac{R_{ij}}{R_l} \qquad R_{ij} < R_l \tag{6}$$

$$\frac{R_{ij}-R_l}{R_h-R_l} + \alpha \qquad R_l \leq R_{ij} \leq R_h \tag{7}$$

$$\frac{R_{ij}}{R_{max}} + \beta \qquad R_{ij} > R_h \tag{8}$$

For values with low tendency

$$\frac{R_h}{R_{ij}} \qquad R_{ij} > R_h \tag{9}$$

$$\frac{R_h-R_{ij}}{R_h-R_l} + \alpha \qquad R_l \leq R_{ij} \leq R_h \tag{10}$$

$$\frac{R_{min}}{R_{ij}} + \beta \qquad R_{ij} < R_l \tag{11}$$

Where,R_{ij}, is the value of i^{th} property of j^{th} service. R_l, is the lower limit of consumer requirement for an attribute, while R_h, is the higher limit of consumer requirement for the attribute. R_{max}, is the maximum value of a QoS property offered by the web services under consideration, and R_{min}, is the minimum value of a QoS property offered by the web services under consideration.In the equations the value of α and $\beta \in \{1, 2, 3, ...\} \quad where \quad \alpha < \beta$.

The calculation of the accuracy matrix is dependent on the tendency of QoS parameters. Tendency explains how the numeric value of a service changes for the service to be perceived as better. The tendency for the majority of values is expected to be high, others are expected to be low such as Response time and Latency. The tendency of a parameter indicates whether high or low values are regarded as more desirable in an ideal scenario. For example, high Availability and Throughput values imply a better service whereas low Response time and Latency values would also generally indicate better service.

The equations 6-11 generate results which are normalized in the range $[0, 1]$. The constants α and β are introduced in order to discriminate between the three ranges. For results in the loose range, the value remains in the range of $[0, 1]$. For results in the preferable range, α is added to the equation and the results are in the range $[\alpha, \alpha + 1]$. For results in the tight range, β is added and the results are in the range$[\beta, \beta + 1]$.

For consistency, using equation (1-6) values of QoS properties were normalized to be in a small range, in which all the values in the accuracy matrix lies in the range of $[0, \beta + 1]$. In the selection process there are many services, the main purpose of the algorithm is to arrange all the QoS values for services based on minimum and maximum values , and arrange the values between $[0, \beta + 1]$.

On the other hand, in the algorithm, every value is considered precisely based on the range. Consider a value for availability; assume the preferable range is $[80\% - 90\%]$ .Any service having availability value less than 80% will be counted as loose, between $[80\% - 90\%]$ will be counted as preferable range and α will be added, finally any value larger than 90% will be counted as tight and β will be added. Since $0 < \alpha < \beta$ it guarantees that the higher range always has higher value than the other two ranges. The results of the calculations are used to populate the accuracy matrix. The matrix shows how accurately each advertised service matches the overall consumer requirement without yet considering the desired preference of each attribute. After the matrix is fully populated, the rank of each service is calculated by summation of the QoS attribute and weight product for that service using equation (12).

$$\mathrm{R}_{\ i} = \sum_{j=1}^{n} A_{ij} * W_j \tag{12}$$

Where R_i represents rank of i^{th} service , A_{ij} represents the accuracy value of j^{th} QoS property of i^{th} service, and W_j represents weight of j^{th} QoS property.

IV. SIMULATION ENVIRONMENT

A. Performance metrics

Performance was evaluated with the following measures: Recall: the ratio of the number of found files to the number of all matched files in the network.

Average path length (APL) of searches: the average distance from the query originator to the targeted node which first finds a matched file. If none are found, the average path length of the search is set as four $[TTL + 1]$ in the simulations. This metric is used to measure the speed of resource discovery. Other metrics are Recall per visited node, Average number of found files, Number of visited nodes, Number of query messages.

B. Topology initialization and evolution

In order to better observe the evolution of network topology in the simulations, the process was started from a small-size random network with 100 nodes. Each peer node connected to four peer nodes bi-directionally based on similarity to generate a homophily topology. Since at the initial phase there is no interaction between peer nodes, each peer node has an empty knowledge index which can contain a maximum of 1000 entries between topics and associated peer nodes (if no other size is specified). In the simulations, a dynamic network was started with a small set of peer nodes. The peer nodes frequently transition between being present or absent from the network. In order to explore the effect of topology, random topology, ring topology and power law topology are also implemented to draw comparisons regarding the efficiency of service discovery.

C. Content generation and distribution

In the experiments, each network was initialized with 100 nodes. Each node offers 11 semantic web services which are uniformly distributed to the node. All the experiments were performed with real semantic services. The set of semantic services used for the experiments were from the test collection OWL-S TC4 [1]. Based on the OWL-S TC4 dataset, which has 1082 semantic web services, the semantic content of the services including ID, Name, Input, Output and discretion are extracted for the process of service discovery. Some key QoS parameters such as response time, availability, reliability, latency are created randomly and assigned to the services for the process of service selection.

D. Search network

In each time step of the simulations, an online node was randomly chosen as the requesting node and a search with a query was started. Queries were uniformly generated among all the agents. This means that all the agents in the system had the same probability of generating service queries. Each query was tagged with a TTL to limit the life time of a message to three hops in the simulations. The generated queries were propagated with the proposed routing algorithm. To decide the average degree of connection it is influence on the average path length was evaluated. As the average degree of connection increases, the paths get shorter and it is easier to locate the target agent since agents have available more possibilities to guide the search.

V. PERFORMANCE EVALUATION

A. Service discovery performance

In this section results for precision and recall were compared for the homophily topology against random, ring and power law topologies. In each case, a node was randomly chosen to be the requesting node from the 100 node vector and a search was started with a query selected randomly from the list of queries in the current dataset. Each query is tagged by a TTL to limit the lifetime of a message to three hops in our simulation. The selected queries are propagated with the Gnutella routing algorithm.

For the evaluation 24 queries were selected randomly from the list of queries in the dataset, for each query the simulations of Service discovery were run. The tests were repeated for different TTL values starting from 2 to 11.For performance measurement we used precision and recall. Precision and recall are standard measurements that have been used in information retrieval for measuring the accuracy of a discovery or recommendation method or a discovery engine. Precision is

[1]http://www.semwebcentral.org/projects/owls-tc

defined as the ratio of the number of returned correct services to the total number of all returned services [21]. By contrast, recall is defined as the ratio of the number of returned correct services to the number of all correct services.

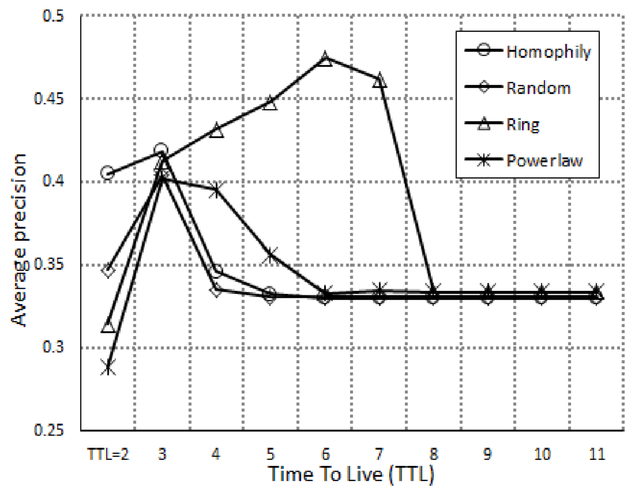

Fig. 3. Average Precision calculation for Homophily, Ring, Power Law and Random topologies

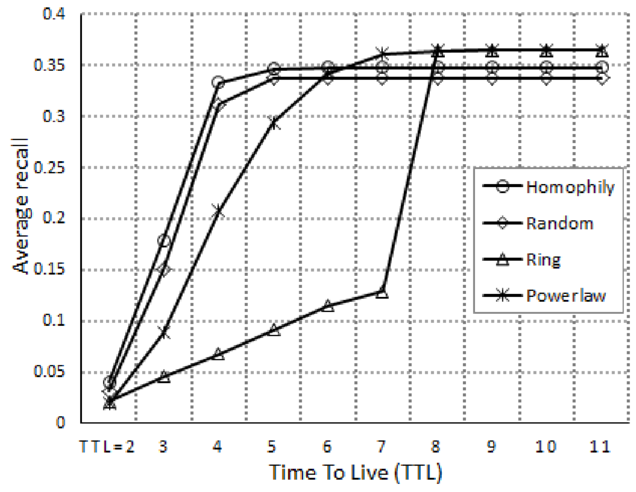

Fig. 4. Average Recall calculation for Homophily, Ring, Power Law and Random topologies

From the results shown in fig. 3 and fig. 4 it is evident that homophily topology achieved better performance for lower TTL values. Increasing TTL values means increasing in the message exchange between nodes which in result increases overall system overload. For average precision homophily topology outperformed other topology for TTL values of 2, and 3. On the other hand for average recall homophily topology again outperformed other topologies for TTL values 2,3,4,5, and 6. If we consider values of both recall and precision, for example homophily topology has best precision (0.418) with TTL value 3, within the same TTL it has the best recall (0.179) comparing to other topologies as well. But for ring topology it has best value (0.474) for TTL value 6, while it is recall value (0.114) for the same TTL is the worst among all the topologies. It is the same for Powerlaw topology; it is best precision value (0.395) is for TTL 4 while it is recall for the same TTL is the second worst after ring topology with the value of (0.2). From the results we can see that homophily gets better precision and recall for lower TTL values and for the higher TTL the average of recall and precision is better than the other topologies.

B. *Service selection performance*

In the proposed algorithm the rank of web services was calculated based on QoS properties. To measure the performance the algorithm was compared to proposed algorithm in [1] which used probabilistic flooding-based method, by combining Simple Additive Weighting (SAW) technique and Skyline filtering. And also compared to to SPSE algorithm, which uses Pareto optimal-based solution method [22].

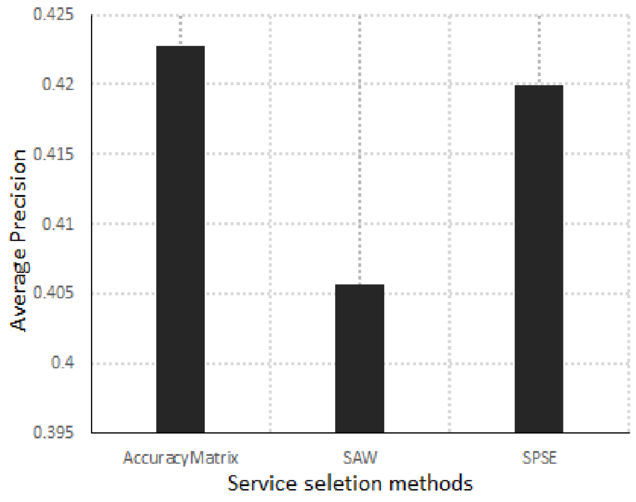

Fig. 5. Average Precision for AccuracyMatrix, SPSE and SAW methods

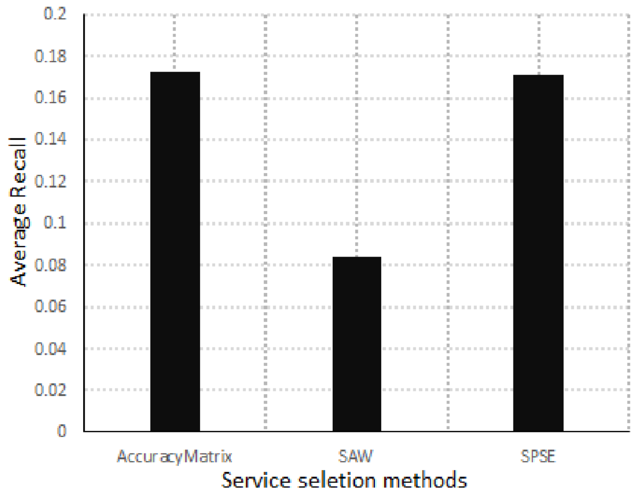

Fig. 6. Average Recall for AccuracyMatrix, SPSE and SAW methods

For the selection process the same set of tests were used as in the discovery process. Since the semantic dataset does not have any QoS properties attached to it, we randomly added four QoS properties to services in the dataset. The added QoS properties were Response Time, Availability, Reliability and Latency. For each test different requirements were used for QoS properties. Based on the requirements we checked number of returned services and relevant returned services.

From fig. 5 and fig. 6 it is apparent that the Accuracy Matrix approach improved both precision and recall compared to SAW and SPSE, the percentage of returned services in the Accuracy Matrix is higher than SAW and SPSE. In the same way the percentage of returned relevant services in Accuracy Matrix is higher than both SAW and SPSE. The improvement is due to the inclusion of partially matched web services in the Accuracy Matrix approach. This confirms that our approach deals more accurately with the service query, and returns relevant service to the consumer even in the case where the candidate services only partially match the consumer requirements.

VI. Conclusion

In this work, we proposed a context-aware service discovery and selection method in a decentralized environment. In the approach, each node in the network established connection to the other nodes based on homophily similarity between them. For the homophily a cosine similarity method was used which calculates service similarity based on inputs, outputs and categories of services. With cosine homophily nodes create a network based on similarity without having central supervision. Cosine similarity is used for bootstrapping and creation of network community, therefore it does not require any training period. With an increasing number of interactions between nodes the knowledge index gathers information about content of neighbouring nodes which in result helps the process of service discovery. In the experiment we compared Cosine homophily topology to Ring, Power Law, and Random topologies, the results showed improvements in both precision and recall in the service discovery. This is due to the use of homophily similarity which let the nodes to create community based on similarity of their contents.

One of the other features of this paper is the inclusion of partially matching services. When an initial list of web services is returned to the requesting node the algorithm includes them in the selection process. The algorithm considers every service accurately and ranks them based on QoS properties and node knowledge. Experimental results show that Accuracy Matrix performed better comparing to SAW algorithm. Accuracy Matrix had better precision and recall comparing to SAW.

Acknowledgment

This work was partially supported by the National Natural the Natural Science Foundation of Jiangsu Province under Grant No.BK20130528 and Natural Science Foundation of China under Grants No.61502209 and 61202474.

References

[1] W. Lin, W. Dou, Z. Xu, and J. Chen, *A QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network*, Concurrency and computation, vol. 25, no. 13, pp. 18431860, February 2013.

[2] E. del Val, M. Rebollo, and V. Botti, *Enhancing decentralized service discovery in open service-oriented multi-agent systems*, Autonomous Agents and Multi-Agent Systems , vol. 28, no. 1, pp. 1-30 , January 2014.

[3] D. Skoutas, D. Sacharidis, V. Kantere, and T. Sellis, *Efficient Semantic Web Service Discovery in Centralized and P2P Environments*, in The Semantic Web - ISWC 2008.: Springer Berlin Heidelberg, 2008, ch. 37, pp. 583-598.

[4] M. Ahmed, Lu Liu, J. Hardy, and B. Yuan, *An Efficient Algorithm for Partially Matched Web Services Based on Consumer's QoS Requirements*, in IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), London, 2014, pp. 859-864.

[5] M. Klusch, B. Fries, and K. Sycara, *Automated semantic web service discovery with OWLS-MX*, in AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, New York, USA, 2006, pp. 915 - 922.

[6] K. Kritikos , D. Plexousakis, *Novel Optimal and Scalable Nonfunctional Service Matchmaking Techniques*, Services Computing, IEEE Transactions on , vol.7, no.4, pp.614,627, Oct.-Dec. 2014

[7] Y. Chen, J. Huang, and C. Lin, *Partial Selection: An Efficient Approach for QoS-Aware Web Service Composition*, Web Services (ICWS), 2014 IEEE International Conference on , vol., no., pp.1,8, Jun.-Jul. 2014

[8] D. Bachlechner, K. Siorpaes, D. Fensel, and I. Toma, *Web service discovery - a reality check*, in In Proceedings of the 3rd European Semantic Web Conference, 2006.

[9] S. Agarwal and R. Studer, *Automatic matchmaking of Web services*, in in Proceedings of the IEEE International Conference on Web Services, 2006, pp. 4554.

[10] P. Plebani and B. Pernici, *URBE:Web ServiceRetrieval Based on similarity evaluation*, IEEE Trans. Know. Data Eng., vol. 21, no. 11, pp. 1629-1642, November 2009.

[11] M. Klusch, B. Fries, and K. Sycara, *OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services*, Web Seman., Sci., Serv. AgentsWorld Wide Web, vol. 7, no. 2, pp. 121-133, April 2009.

[12] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin, *A Decentralized Service Discovery Approach on Peer-to-Peer Networks*, IEEE TRANSACTIONS ON SERVICES COMPUTING, vol. 6, no. 1, pp. 64-75, January-March 2013.

[13] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, in SIGCOMM '01 Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, USA, 2001, pp. 149-160.

[14] A. Rowstron and P. Druschel, *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*, in IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany, 2001, pp. 329-350.

[15] P. Maymounkov and D. Mazires, *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*, in First InternationalWorkshop, IPTPS, Cambridge, MA, USA, 2002, pp. 53-65.

[16] M. McPherson, L. Smith-lovin, J. Cook, *Homophily in social networks*, birds of a feather, Annual review of sociology, 201.

[17] Antony I. T., Rowstron and P. Druschel, *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems*, in In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg (Middleware '01), London, 2001, pp. 329-350.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*, in In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01), New York, 2001, pp. 161-172.

[19] P. Lazarsfeld, *Friendship as a social process: A substantive and methodological analysis*, Freedom and control in Modern Society, 1954.

[20] J. Ye, *Cosine Similarity Measures for Intuitionistic Fuzzy Sets and their Applications*, Mathematical and Computer Modelling, vol. 53, no. 1-2, pp. 91-97, January 2011.

[21] S. Dimitrios, S. Dimitris, K. Verena, and S. Timos, *Efficient Semantic Web Service Discovery in Centralized and P2P Environments*, in The Semantic Web - ISWC 2008.: Springer Berlin Heidelberg, 2008, ch. 37, pp. 583-598.

[22] L. Zhao, Y. Ren, M. Li, and K. Sakurai, *Flexible service selection with user-specific QoS support in service-oriented architecture*, Network and Computer Applications, vol. 35, no. 3, pp. 962-973, May 2012.