Towards P2P-based Semantic Web Service Discovery with QoS Support *

Le-Hung Vu, Manfred Hauswirth and Karl Aberer

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{lehung.vu, manfred.hauswirth, karl.aberer}@epfl.ch

Abstract. The growing number of web services advocates distributed discovery infrastructures which are semantics-enabled and support quality of service (QoS). In this paper, we introduce a novel approach for semantic discovery of web services in P2P-based registries taking into account QoS characteristics. We distribute (semantic) service advertisements among available registries such that it is possible to quickly identify the repositories containing the best probable matching services. Additionally, we represent the information relevant for the discovery process using Bloom filters and pre-computed matching information such that search efforts are minimized when querying for services with a certain functional/QoS profile. Query results can be ranked and users can provide feedbacks on the actual QoS provided by a service. To evaluate the credibility of these user reports when predicting service quality, we include a robust trust and reputation management mechanism.

1 Introduction

The increasing number of web services demands for an accurate, scalable, effective and reliable solution to look up and select the most appropriate services for the requirements of the users. This is specifically complicated if numerous services from various providers exist, all claiming to fulfill users' needs. To solve these problems, a system basically has to provide expressive semantic means for describing web services including functional and non-functional properties such as quality of service (QoS), semantic search capabilities to search distributed registries for services with a certain functional and QoS profile, and mechanisms for allowing users to provide feedbacks on the perceived QoS of a service that can be evaluated by the system regarding their trustworthiness.

In this paper we present our approach to address these issues. It is based on requirements from a real-world case study of virtual Internet service providers

* The work presented in this paper was (partly) carried out in the framework of the EPFL Center for Global Computing and was supported by the Swiss National Funding Agency OFES as part of the European project DIP (Data, Information, and Process Integration with Semantic Web Services) No 507483. Le-Hung Vu is supported by a scholarship of the Swiss federal government for foreign students.

(VISP) in one of our projects[1]. In a nutshell, the idea behind the VISP business model is that Internet Service Providers (ISPs) describe their services as semantic web services, including QoS such as availability, acceptable response time, throughput, etc., and a company interested in providing Internet access, i.e., becoming a VISP, can look for its desired combination of services taking into account its QoS and budgeting requirements, and combine them into a new (virtual) product which can then be sold on the market. At the moment this business model exists, but is done completely manually.

Since many ISPs can provide the basic services at different levels and with various pricing models, dishonest providers could claim arbitrary QoS properties to attract interested parties. The standard way to prevent this is to allow users of the service to evaluate a service and provide feedbacks. However, the feedback mechanism has to ensure that false ratings, for example, badmouthing about a competitor's service or pushing own rating level by fake reports or collusion with other malicious parties, can be detected and dealt with. Consequently, a good service discovery engine would have to take into account not only the functional suitability of the services but also their prospective quality offered to end-users regarding to the trustworthiness of both providers and consumer reports. According to several empirical studies [23, 28], this issue of evaluating the credibility of user reports is one of the essential problems to be solved in the e-Business application area.

To achieve the high scalability, in our work we focus on developing a decentralized discovery approach and for improved efficiency we use a structured overlay network as the decentralized service repository system. In the following we assume that web services are being described semantically including QoS properties, for example, using WSMO[2], descriptions can be stored in distributed registries, and users can provide feedbacks on the experienced QoS. Based on these realistic assumptions we will devise a framework for P2P-based distributed service discovery with QoS support.

Regarding the semantic characterization of Web Services several properties can be considered, of which the most obvious are the structural properties of the service interface, i.e., the input and output parameters of a service. Another important aspect, in particular for distinguishing services with equivalent functional properties, relates to QoS characteristics. In our approach we intend to support both aspects. As described above, for QoS it is of interest to compare the announced with the actual service performance, for which we take a reputation-based trust management approach. Other characteristics of Web Services, in particular the process structure of the service invocation also have been considered, e.g., Emekci et al [15], but we consider these as less important, since they are difficult to use in queries and unlikely to be the primary selection condition in searches, and thus not critical in terms of indexing. However, we may expect that the service interface will be usually used as a search condition with good selectivity among a large number of web services. In order to support these queries

[1] http://dip.semanticweb.org/
[2] http://www.wmso.org/

we have to index unordered key sets (corresponding to a service interface), where the keys are usually taken from a (shared) domain ontology. To the best of our knowledge, although the issue of indexing semantic data in structured overlay networks has already been mentioned somewhere, e.g., [7, 13, 33], none of them have taken into account the structural properties of web services while indexing semantic service descriptions for the benefits of service discovery.

The major contribution of this paper is the proposal of a new distributed service discovery framework which is expected to be scalable, efficient and reliable. With the use of structured peer-to-peer overlays as the service repository network, the system is highly scalable in terms of number of registries and services. Our approach uses multiple unordered key sets as index terms for semantic web service descriptions, thus make it possible to quickly identify the registries containing most likely matched services according to user requests. The local semantic service matchmaking at a specific registry can also be performed efficiently thanks to the combination of the ontology numerical encoding scheme [9] with the pre-computation of the matching information between service advertisements and possible user queries [32] to reduce the time-consuming reasoning steps. In addition, the search algorithm exploits the generalization hierarchy of the underlying ontology for approximate matching and will use QoS information to rank the search results according to preferences of users. Our QoS-based service selection and ranking algorithm also takes into account the issue of trust and reputation management sufficiently, thereby returning only the most accurate and relevant results w.r.t. user requirements.

2 Related Work

Our framework uses a novel ontology-based approach to distribute service advertisements appropriately among a P2P network of registries. This method is different from that of METEOR-S [35] and HyperCup [29] as we do not base it on a classification system expressed in service or registry ontologies. In these approaches, the choosing of a specific registry to store and search for a service advertisement depends on the type of the service, e.g., business registry is used for storing information of business-related services. In fact, these proposals is good in terms of organizing registries to benefit service management rather than for the service discovery itself. Although publishing and updating service description information based on their categories is relatively simple, it would be difficult for users to search for certain services without knowing details of this classification, and it would be hard to come up with such a common service or registry ontology. To some extent our approach is similar to WSPDS [18], but our methods are specifically targeted at *structured* P2P overlay networks in order to support more efficient service publishing and discovery. We use our P-Grid P2P system [2] as the underlying infrastructure, which at the time of this writing, is among the very few P2P systems which support maintenance and updating of stored data. [30] indexes service description files (WSDL files) by a set of keywords and uses a Hilbert-Space Filling Curve to map the n-dimensional

service representation space to an one-dimensional indexing space and hash it onto the underlying DHT-based storage system. However, the issue of characterizing a semantic service description as a multi-key query in order to support semantic discovery of services has not yet been mentioned in this work. As aforementioned, Emekci et al [15] suggest to search services based on their execution paths expressed as finite path automata which we consider less important since this is difficult to use as primary selection condition in queries as user would need to know and describe the execution flow of their required services.

Regarding QoS, although the traditional UDDI registry model [1] does not refer to quality of web services, many proposals have been devised to extended the original model and describe web services' QoS capabilities, e.g., QML, WSLA and WSOL [14, 20]. The issue of trust and reputation management in Internet-based applications as well as in P2P systems has also been a well-studied problem [12, 16]. However, current QoS provisioning models have not sufficiently considered the problem of evaluating the credibility of reporting users. The existing approaches either ignore this issue totally [4, 8, 17, 34] or employ simple methods which are not robust against various cheating behaviors [15, 19]. Consequently, the quality of ranking results of those systems will not be assured if there are dishonest users trying to boost the quality of their own services and badmouthing about the others. [11] suggests augmenting service clients with QoS monitoring, analysis and selection capabilities. This is a bit unrealistic as each service consumer would have to take the heavy processing role of both a discovery and a reputation system. Other solutions [22, 24, 26, 27] use mainly third-party service brokers or specialized monitoring agents to collect performance of all available services in registries, which would be expensive in reality.

An advanced feature of our architecture is that we perform the service discovery, selection and ranking based on the matching level of service advertisements to user queries both in terms of functionality and QoS as well as taking into account trust and reputation adequately. Our QoS provisioning model is developed from [8, 17, 19] using concepts of integrating QoS into service description by [27] and [34]. The trust and reputation management mechanism originally combines and extends ideas of [3, 10, 21, 37, 38] and is the first solution to address the most important issues sufficiently.

3 A Model for P2P-based Web Service Discovery with with QoS Support

Fig. 1 shows the conceptual model of our distributed service discovery framework.

Service advertisements with embedded QoS information are published in P2P-based registries by various providers (1), and users can query for services with certain functionalities and required QoS levels (2) using any registry peer as their access point. The P2P-based registries then take care of routing the request to the peer(s) that can answer it (3). The results will be returned to the user (4) and this user may invoke one of the found services. Additionally,

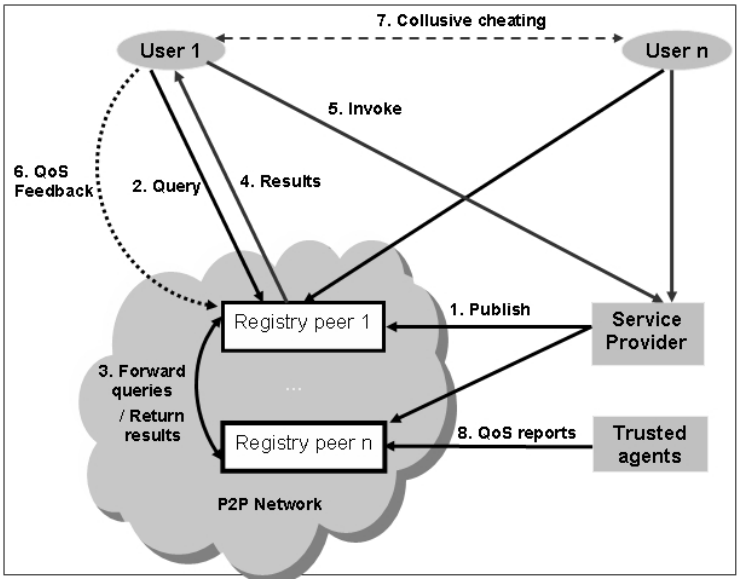

Fig. 1. Framework model

users can express feedbacks on the QoS they could obtain from a service to the registry peers managing that service (6).

The evaluation of QoS reports by the registry peers has to account for malicious reporting and collusive cheating of users (7) to get a correct view of the QoS properties of a service. Additionally, we also allow trusted agents in the model to provide QoS monitoring for certain services in the system (8). These well-known trusted agents always produce credible QoS reports and are used as trustworthy information sources to evaluate the behaviors of the other users. In reality, companies managing the service searching engines can deploy special applications themselves to obtain their own experience on QoS of some specific web services. Alternatively, they can also hire third party companies to do these QoS monitoring tasks for them. In contrast to other models [22, 24, 26, 27, 34] we do not deploy these agents to collect performance data of all available services in the registries. Instead, we only use a small number of them to monitor QoS of some selected services because such special agents are usually costly to setup and maintain.

Fig. 2 shows the internal architecture of a registry peer.

The communication module provides an information bus to connect the other internal components; interacts with external parties, i.e., users, trusted agents, and service providers, to get service advertisements, QoS data, and feedbacks; and provides this information to the internal components. Additionally, it is the registry peer's interface to other registry peers (query forwarding, exchange of service registrations and QoS data) and for the user to submit queries and receive

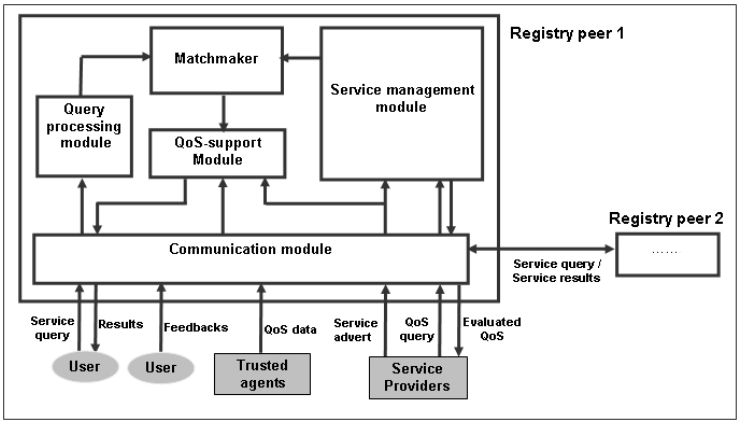

Fig. 2. Registry Peer Structure

results. The query processing module analyzes a semantic web service query into user's required functionality and the corresponding QoS demand of the needed service and then forwards them to the matchmaker module. The matchmaker compares the functional requirements specified in a query with the available advertisements from the service management module to select the best matching services in terms of functionality. The list of these services is then sent to the QoS support module, which performs the service selection and ranking, based on QoS information provided in the service advertisements and QoS feedback data reported by the users, so that the result contains the most relevant web services according to user request. Providers are also able to query the evaluated QoS of their own services and decide whether they should improve their services' performance or not.

4 Service Description, Registration, and Discovery

A semantic service description structure stored in a peer registry includes:

- a WSDL specification of the service.
- service functional semantics in terms of service inputs, outputs, pre-conditions, post-conditions and effects, which is described by WSMO ontology concepts using the techniques proposed by [31].
- optional QoS information with the promised QoS for the service.

During operation of the system this information will be matched against semantic queries which consist of:

- functional requirements of user in terms of service inputs, outputs, pre-conditions, post-conditions and effects, also expressed in WSMO concepts.

– optional user's QoS requirements provided as a list of triples $\{q_i, n_i, v_i\}$, where q_i is the required QoS parameter, n_i is the order of importance of q_i in the query (as user preference) and v_i is the user's minimal required value for this attribute.

Quality properties of web services are described by concepts from a QoS ontology and then embedded into the service description file using techniques suggested by Ran [27] and WS-QoS [34]. In our work, the value of a quality parameter of a web service is supposed to be *normalized* to a non-negative real-valued number regarding service-specific and call-specific context information where higher normalized values represent higher levels of service performance. For instance, a web service with a normalized QoS parameter value for reliability of 0.99 will be considered as more reliable to another one with a normalized reliability value of 0.90. In this case the normalized reliability is measured as its degree of being capable of maintaining the service and service quality over a time period T. Another example would be a web service with a normalized execution-time value 0.50 which will be considered as faster than another service with a normalized execution-time value 0.20. In this case the normalized value is computed as the reciprocity of the execution-time w.r.t. some client-dependent conditions. For experimental evaluations, we have developed a QoS ontology for the VISP use-case using WSMO. This QoS ontology includes the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, price, etc. We currently assume that users and providers share a common ontology to describe various QoS concepts. However, this could be relaxed with the help of many existing ontology mapping frameworks. The QoS provisioning model is described in details in [36].

4.1 A Closer Look at Semantic Service Descriptions

In our architecture, a semantic service description, i.e., a service advertisement or a service query, will be associated with a multi-key vector, which we call the *the characteristic vector* of the service. Based on this vector service advertisements are assigned to peer registries. Similarly, discovery of registries containing services relevant to a user query is also based on the characteristic vector of the query itself.

First, all ontological concepts representing *inputs* and *outputs* of a web service advertisement/service request will be categorized into different *Concept Groups* based on their semantic similarity. This semantic distance between two concepts is computed based on the distance between them in the ontology graph and their number of common properties as proposed by previous work, e.g., [6]. Each group has a root concept defined as the one with the highest level in the ontology graph, i.e., the most general concept, among all member concepts.

A semantic service description, i.e., a service advertisement or a service query, is then characterized by the concept groups that its input and output parameters belong to. According to [9], ontological concepts can be mapped onto numerical key values in order to support semantic reasoning efficiently. Therefore, we can

utilize keys to represent concepts in our work. A group of similar concepts is then associated with a Bloom key built by applying k hash functions $h_1, h_2, \cdots, h_k$ to the key of each concept member, allowing us to quickly check the membership of any concept to that group [5]. For each input I_i (or output O_i) of a service, we firstly find the concept group CG_i that it belongs to. As the order of inputs/outputs of a service generally has no sense in determining its functionality, we define a total ordering of various concept groups as in Definition 1 so that service queries/advertisements with similar interfaces would have the same characteristic vector regardless the differences in the order of their parameters. *The characteristic vector* of this service description is then represented by the list of corresponding Bloom keys of all CG_is, sorted in the descending order of CG_i.

Definition 1. A concept group CG_x is considered as having *higher order* ($>$) than another group CG_y if:

1. The level of CG_x in the ontology graph is higher than the level of CG_y or:
2. Both CG_x and CG_y have the same level and CG_x is in the left of CG_y in the ontology graph.

The partitioning of ontological concepts is illustrated in Fig. 3 where C_j is an ontological concept and CG_i is a concept group. The task of fragmenting the ontology graph is similar to that of relational and semi-structured database systems, which could be performed semi-automatically by the system with additional user support.

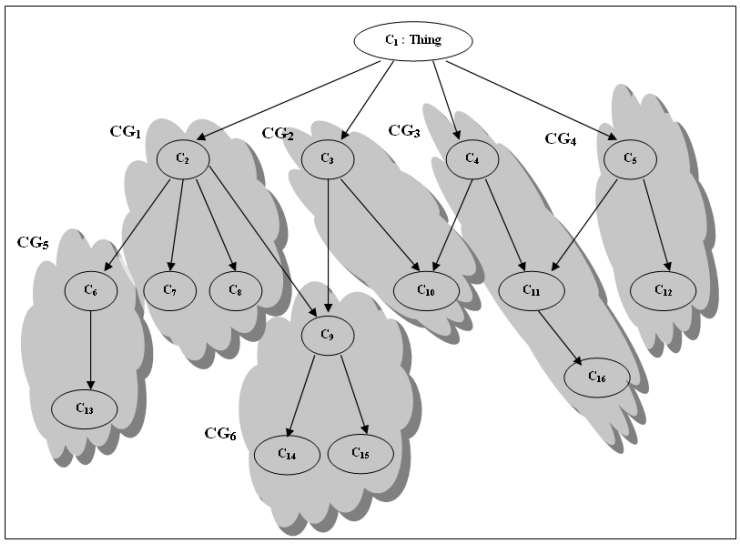

Fig. 3. Ontology graph partitioning

The root concepts of CG_1, CG_2, CG_3, CG_4, CG_5 and CG_6 are C_2, C_3, C_4, C_5, C_6 and C_9, respectively. The total ordering of all concept groups is $CG_1 > CG_2 > CG_3 > CG_4 > CG_5 > CG_6$. As an example, let us assume that we have a service description S_1 with inputs C_7, C_{14}, C_{10} and outputs C_{12}, C_{16} which belong to concept groups CG_1, CG_6, CG_2 and CG_4, CG_3, respectively. Regarding the above ordering relation, this service description is then represented by *the characteristic vector* $V = \{k_1, k_2, k_6, k_d, k_3, k_4\}$, where k_i is CG_i's Bloom key and k_d is a dump value to separate S_1's inputs and outputs.

Although we are using only inputs and outputs of a service in its multiple-key representation, we believe that the extension of this idea to other features in a semantic service description, e.g., pre-conditions, post-conditions, effects, could be done in a similar fashion. The strategy used for partitioning the ontological graph will not affect the correctness but mainly the efficiency of the discovery algorithm. For instance, although it is tempting to allow a concept to belong to more than one group while partitioning, this increases the discovery time because we need to contact different registries to search for all possibly matching services. Therefore, we prefer to have only one group for each concept. For simplicity, we currently assume that all registries agree on one ontology of concepts, but this restriction will be relaxed soon with our on-going work.

4.2 Mapping of Service Advertisements to Registries

Each registry peer is responsible for managing certain web services that operate on a certain set of concepts. The mechanism to assign these sets to peers works as follows:

1. Each vector $V_i = \{k_{i1}, k_{i2}, \ldots, k_{in}\}$, where k_{ij} $(j = \overline{1..n})$ is a group's Bloom key or dump value k_d, is mapped to a combined key K_i using a special function H_c that includes all features of each individual member key k_{ij}.
2. Using the existing DHT-based searching mechanism of the underlying P-Grid network [2], we can easily find the identifier RP_i of the registry peer that corresponds to the key K_i.
3. The registry peer RP_i is responsible for storing the description of those services with the same characteristic vector V_i.

This assignment of services to registries during the publishing phase will help us to quickly identify the registry(-ies) most likely to contain a semantic description matching with a service request in the later discovery process. Using Bloom filters, the step of checking the membership of a concept in certain concept groups can be done fast and with very high accuracy level. Therefore, the computation of the characteristic vector of a service request can be done efficiently. Eventually, the question of searching for the registry(-ies) most likely to store a matched services becomes the problem of finding the peers capable of answering a multi-keyword query which corresponds to this characteristic vector in the P2P network. This problem can be solved by using one of the two following approaches. The first one is to simply concatenate all k_{ij}s together and then

use this as the index/search key in the underlying P2P network. The second possibility is to deploy another type of peers in the network as *index peers* to keep identifiers of those registries that manage keywords related to various combination of k_{ij}s. Of course, there is another naive method in which we can search for all peers storing each concept term and then intersect all partial matches to get the final results. However, we reason that this approach would be inefficient due to the following reason. As the semantics of the parameters in a service interface are generally different from each other, a registry containing service advertisements with only one satisfactory parameters does not necessarily store service descriptions with the full interface as user requires. This means it would be costly to forward the service query to many (distributed) registries and wait for all semantic matchmaking in these repositories to terminate and get the final results.

We have decided to use the first method because in this way, the keyword generating function H_c will generate similar keys K_is for services with similar characteristic vectors $\{k_{i1}, k_{i2}, \ldots, k_{in}\}$. Since P-Grid uses *prefix-based query routing* as its search strategy, services corresponding to similar K_is, which are likely to offer comparable functionalities, will be assigned to registries adjacent to each other (P-Grid clusters related information). This is important as with the very high number of registries and published services, the query for services will only need to be forwarded to a small number of adjacent peers. Otherwise, we will have to wait for the results to be collected from a lots of widely distributed registries, making the searches become highly inefficient. Moreover, this is advantageous for the exchanges of QoS reports and user reputation information among neighboring registries later during the QoS predicting process.

Regarding Fig. 3, supposed that we have three services: S_1 operating on two concepts C_2, C_3 and producing C_4, S_2 operating on two concepts C_2, C_9 and producing C_{14}, S_3 operating on two concepts C_2, C_9 and producing C_{15}. The characteristic vectors of S_1 will be $\{k_1, k_2, k_d, k_3\}$ whereas S_2, S_3 will have the same characteristic vector as $\{k_1, k_6, k_d, k_6\}$, with k_1, k_2, k_3, k_6 is the Bloom key of the concept groups CG_1, CG_2, CG_3, CG_6 and k_d is a dump key, respectively. According to our way of distributing service descriptions, S_1 will be assigned to one registry peer P_1 with index key $K_1 = k_1 \| k_2 \| k_d \| k_3$ and S_2, S_3 will be assigned to another peer P_2 with another index entry $K_2 = k_1 \| k_6 \| k_d \| k_6$.

4.3 Pre-computation of Service Matching Information to Support Semantic Service Discovery

Since the publishing task usually happens once and is not a computationally intensive process, we can devote more time in this stage to reduce later discovery time, as suggested by Srinivasan et al [32]. However, their proposed approach is not scalable since it requires to store the matching information of all services which match each concept c_i in the ontology, thus producing much redundant information. Hence, we improve their method by observing that if a concept c_i of a group CG_i, is similar to another concept c_j (also belonging to this group),

then both of them should have approximately the same distance, i.e., the same level of semantic similarity, to the root concept of CG_i.

Accordingly, for each CG_i, we store a matching list containing semantic distances from each parameter of each service to CG_i's root concept. For example, assuming that we have a registry peer responsible for managing those services which operate on the list of concept groups CG_1, CG_2,..., CG_k. Then in the matching table of this registry, we store for each group CG_i, $i = \overline{1..k}$, a list $Ldst_i$ of records $\{[S_{i1}, d_1], [S_{i2}, d_2], \cdots, [S_{in}, d_n]\}$, where S_{ij} represents a web service, $d_j \in [0, 1]$ is the semantic similarity between the concept represented by one parameter of S_{ij} with the root concept of CG_i, $j = \overline{1..n}$, n is the number of services in this registry.

A query for a service is first submitted to a registry peer. At this entry point the characteristic vector of the query is computed as in Section 4.1 and Section 4.2. Using the combined key of this characteristic vector as a search key, the query is then forwarded by P-Grid's routing strategy to a registry most possibly containing matching services. For each service query's parameter c_i belonging to group CG_i, the discovery algorithm at this registry computes its matching level d_i with CG_i's root concept rc_i. Afterward, it finds the list L_i of those services S_{ij}s each of which has (at least) one parameter with an approximate matching level d_{ij} with rc_i, i.e., $d_{ij} \approx d_i$, by browsing the matching list $Ldst_i$ of each rc_i. We then intersect all L_is to get the list L_c of possibly matching services. Note that if c_{i1} and c_{i2} have the same matching level d_i with CG_i's root concept, we can only conclude that c_{i1} and c_{i2} are *possibly* similar. Consequently, simply intersecting all L_is does not help us in finding the services which accurately match the query as in [32]. However, they do allow us to select the list of all possible matches and filter out non-related services, which really reduces the searching time in case the number of registered services is high. Finally, we utilize another service semantic matchmaking algorithm, e.g. [25], to further select from L_c the list L of most suitable services in terms of functionality.

For supporting queries with QoS requirements, we use another table to store the matching information for frequently accessed QoS attributes. With each QoS attribute q_j in this QoS matching table, we have a list $Lqos_j$ of records $\{S_{ij}, w_{ij}, predicted_{ij}\}$, in which S_{ij} identifies a service, w_{ij} is a weight assigned to the semantic similarity between q_j and the QoS attribute q_{ij} supported by S_{ij}, and $predicted_{ij}$ is the value of q_{ij} predicted by our QoS-based service selection and ranking engine. Apparently, we only store in $Lqos_j$ information of those S_{ij}s with w_{ij}s greater than a specific threshold. The idea behind is that we will give higher ranks for services which offer the most accurate QoS concepts at the higher levels compared to the ones required by users. Note that although it is possible to use QoS properties as ranking criteria for service queries without explicit QoS requirements, we have not yet employed this in our current study. Therefore, the QoS-based service selection and ranking phase will be performed only if users provide their QoS requirements explicitly in corresponding queries.

Given the list L of services with similar functionalities, the discovery engine performs the QoS-based service selection and ranking as in Algorithm 1.

Algorithm 1 QosSelectionRanking(ServiceList L, ServiceQuery Q)

Derive the list of QoS requirements in Q: $L_q = [q_1, n_1, v_1], ..., [q_s, n_s, v_s]$
Initialize $QosScore[S_i] = 0.0$ for all services in L;
for each quality concept $q_j \in L_q$ **do**
  for each service $S_i \in L$ **do**
    Search the list L_{qos} of q_j for S_i;
    if S_i is found **then**
      $PartialQosScore = w_{ij} \frac{predicted_{ij} - v_j}{v_j}$;
      $QosScore[S_i] = QosScore[S_i] + \frac{n_j}{\sum n_j} PartialQosScore$;
    else
      Remove S_i from L;
    end if
  end for
end for
Return the list L sorted in descending order by $QosScore[S_i]$ s;

To facilitate the discovery of services with QoS information, we must evaluate how well a service can fulfill a user query by predicting its QoS from the service's past performance reported in QoS feedbacks. In our model, we apply time series forecasting techniques to predict the quality values from various information sources. Firstly, we use the QoS values promised by providers in their service advertisements. Secondly, we collect consumers' feedbacks on QoS of every service. Thirdly, we use reports produced by trusted QoS monitoring agents. In order to detect possible frauds in user feedbacks, we use reports of trusted agents as reference values to evaluate behaviors of other users by applying a trust-distrust propagation method and a clustering algorithm. Reports that are considered as incredible will not be used in the predicting process. Through various experiments, this proposed service selection and ranking algorithm is shown to yield very good results under various cheating behaviors of users, which is mainly due to the fact that the use of trusted third parties monitoring QoS of a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments. The detail of this QoS-based service selection and ranking phase as well as various experimental results are presented in [36].

5 Conclusions and Future Work

In this paper we proposed a new P2P-based semantic service discovery approach which uses a natural way of assigning service descriptions to registry peers. Also, we presented a service selection and ranking process based on both functional and QoS properties. In order to support flexible queries we index unordered key sets where the keys are taken from a shared domain ontology. This problem of indexing of web service descriptions in structured overlay networks to support service discovery has not been addressed so far in the literature. The QoS model

includes a user feedback mechanism which is resilient against malicious behaviors through the application of a trust and reputation management technique that allows us to discover all cheating attempts by providers and service users. As we use a P2P system as the underlying infrastructure, our system scales well in terms of number of registries, search efficiency, number of properties in service descriptions, and number of users.

We already implemented the QoS-based service selection and ranking algorithm with trust and reputation evaluation techniques as a QoS support module in our framework. Many experiments were also performed to prove the effectiveness of our trust and reputation approach under various situations. In the next stage, we will implement the matchmaker based on the work initiated by Paolucci et al [25] and the service management module based on the UDDI standard. The existing implementation of the P-Grid system, Gridella[3], is used as the basis for the communication module. The next step would be to extend our model such that registry peers are able to manipulate with heterogeneous and distributed ontologies. Also, it would be beneficial to extend the indexing scheme to include service pre-conditions, post-conditions, effects, etc., in semantic service description structures. Moreover, further work should be done on the use of QoS properties as ranking criteria for service queries without explicit QoS requirements. In addition, we are studying the possibility of developing and utilizing a caching mechanism to exploit the locality and frequency of service usages in the system.

References

1. *Latest UDDI Version (3.0.2), UDDI Spec Technical Committee Draft, Dated 20041019*. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, 2004.
2. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Rec.*, 32(3):29–33, 2003.
3. K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317, New York, NY, USA, 2001. ACM Press.
4. A. S. Bilgin and M. P. Singh. A DAML-based repository for QoS-aware semantic web service selection. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 368, Washington, DC, USA, 2004. IEEE Computer Society.
5. B. H. Bloom. Space/Time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
6. S. Castano, A. Ferrara, S. Montanelli, and G. Racca. Matching techniques for resource discovery in distributed systems using heterogeneous ontology descriptions. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, page 360, Washington, DC, USA, 2004. IEEE Computer Society.

[3] http://www.p-grid.org/Software.html

7. S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli. Helios: a general framework for ontology-based knowledge sharing and evolution in p2p systems. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, page 597, Washington, DC, USA, 2003. IEEE Computer Society.
8. Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung. UX - an architecture providing QoS-aware and federated support for UDDI. In *ICWS '03: Proceedings of the IEEE International Conference on Web Services (ICWS'03)*, 2003.
9. I. Constantinescu and B. Faltings. Efficient matchmaking and directory services. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, page 75, Washington, DC, USA, 2003. IEEE Computer Society.
10. F. Cornelli, E. Damiani, S. C. Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a P2P network. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pages 376–386, New York, NY, USA, 2002. ACM Press.
11. J. Day and R. Deters. Selecting the best web service. In *Proceedings of the IBM Centers for Advanced Study Conference (CASCON '04)*, pages 293–308, 2004.
12. Z. Despotovic and K. Aberer. Possibilities for managing trust in P2P networks. Technical Report IC200484, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, November 2004.
13. H. Ding, I. T. Solvberg, and Y. Lin. A vision on semantic retrieval in p2p network. In *AINA '04: Proceedings of the 18th International Conference on Advanced Information Networking and Applications Volume 2*, page 177, Washington, DC, USA, 2004. IEEE Computer Society.
14. G. Dobson. *Quality of Service in Service-Oriented Architectures*. http://digs.sourceforge.net/papers/qos.html, 2004.
15. F. Emekci, O. D. Sahin, D. Agrawal, and A. E. Abbadi. A peer-to-peer framework for web service discovery with ranking. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 192, Washington, DC, USA, 2004. IEEE Computer Society.
16. A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 2005.
17. S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(user ranking, compliance, verity). In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 200, Washington, DC, USA, 2004. IEEE Computer Society.
18. F. B. Kashani, C.-C. Chen, and C. Shahabi. WSPDS: Web services peer-to-peer discovery service. In *International Conference on Internet Computing*, pages 733–743, 2004.
19. Y. Liu, A. Ngu, and L. Zheng. QoS computation and policing in dynamic web service selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004. ACM Press.
20. H. Ludwig. Web services QoS: External SLAs and internal policies or: How do we deliver what we promise? In *Proceeding of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*, Roma, Italy, 2003. IEEE Computer Society.
21. E. Manavoglu, D. Pavlov, and C. L. Giles. Probabilistic user behavior models. In *Third IEEE International Conference on Data Mining (ICDM)*, pages 203–210, 2003.
22. E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. *SIGecom Exch.*, 3(1):24–31, 2002.

23. M. I. Melnik and J. Alm. Does a seller's e-Commerce reputation matter? Evidence from eBay auctions. *Journal of Industrial Economics*, 50(3):337–349, 2002.
24. M. Ouzzani and A. Bouguettaya. Efficient access to web services. *IEEE Internet Computing*, pages 34–44, March/April 2004.
25. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK, 2002. Springer-Verlag.
26. C. Patel, K. Supekar, and Y. Lee. A QoS oriented framework for adaptive management of web service based workflows. In *Proceeding of Database and Expert Systems 2003 Conference*, pages 826–835, 2003.
27. S. Ran. A model for web services discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.
28. P. Resnick and R. Zeckhauser. Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *The Economics of the Internet and e-Commerce, Advances in Applied Microeconomics*, 11, 2002.
29. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A scalable and ontology-based P2P infrastructure for semantic web services. In *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*, page 104, Washington, DC, USA, 2002. IEEE Computer Society.
30. C. Schmidt and M. Parashar. A peer-to-peer approach to web service discovery. *World Wide Web*, 7(2):211–229, 2004.
31. K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In *Proceedings of the International Conference on Web Services*, 2003.
32. N. Srinivasan, M. Paolucci, and K. P. Sycara. Adding OWL-S to UDDI, implementation and throughput. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, USA, 2004.
33. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 175–186, New York, NY, USA, 2003. ACM Press.
34. M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A concept for QoS integration in web services. In *Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops*, volume 00, pages 149–155, Italy, 2003.
35. K. Verma, K.Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services. *Inf. Tech. and Management*, 6(1):17–39, 2005.
36. H. L. Vu, M. Hauswirth, and K. Aberer. QoS-based service selection and ranking with trust and reputation management. Technical Report IC2005029, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, June 2005.
37. A. Whitby, A. Jøsang, and J. Indulska. Filtering out unfair ratings in Bayesian reputation systems. *The Icfaian Journal of Management Research*, 4(2):48–64, 2005.
38. L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.*, 16(7):843–857, 2004.