EFPL Technical Report IC/2005/014

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) [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, negotiate and provide its service. The VISP then uses this service for its own applications, e.g. creating a new Internet service product for end-users. 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 services but also its prospective quality offered to end-users regarding to the trustworthiness of both providers and consumer reports. According to several recent studies [29, 31], this issue of evaluating the credibility of user reports is one of the essential problems to be solved in the e-Business application area.

In the following we assume that web services are being described semantically including QoS properties, for example, using WSMO [3], 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.

Existing architectures for web service discovery can be categorized along the following dimensions:

The approach used for web service description: Here we can distinguish explicit categorization using a web service ontology, for example, as used by UDDI, but also by METEOR-S [6] and HyperCup [7], from semantic-based description based on service properties, as, e.g., in WSPDS [8]. In our system we follow the second approach.

The approach used for the search architecture: Here we can distinguish approaches based on central directories from distributed or peer-to-peer architectures, as used e.g., in WSPDS. We will devise a distributed approach and for improved efficiency use a structured overlay network approach, in contrast to WSPDS, which employs unstructured overlay networks.

For the semantic characterization of Web Services several properties can be considered. 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 announced with actual performance, for which we take a reputation-based trust management approach.

Other properties of Web Services, in particular the process structure of the service invocation also have been considered, e.g., Emekci et al. [10], but we consider these as less important, since they are difficult to use in queries and they are most likely not used as 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 we have to index un-ordered key sets (corresponding to a service interface), where the keys are usually taken from a (shared) domain ontology. This indexing problem has not yet been addressed in the literature for structured overlay networks. Thus we will propose in this paper an approach for supporting multiple-key sets as index terms in a structured peer-to-peer overlay architecture. 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 user preference.

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 [6] and HyperCup [7] 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. It is relatively simple when publishing and updating service description information based on their categories. However, 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 [8], 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. [9] 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 [10] 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 exactly the execution of their required services.

Although the traditional UDDI registry model [15] does not refer to QoS, many proposals have been devised to extended the original model and describe service quality capabilities, e.g., QML, WSLA and WSOL [16]. The issue of trust and reputation management in Internet-based applications as well as in P2P systems has also been a well-studied problem [17, 18]. 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 [19–22] or employ simple methods which are not robust against various cheating behaviors [10, 27]. 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. [28] 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 registry and a reputation system. Other solutions [23–26] 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 selection and ranking (based on their matching level 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 [20, 22, 27, 31] using concepts of integrating QoS into service description by [19] and [23]. The trust and reputation management mechanism originally combines and extends ideas of [30, 32–35] and is the first solution to address the most important issues adequately.

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, 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). Their

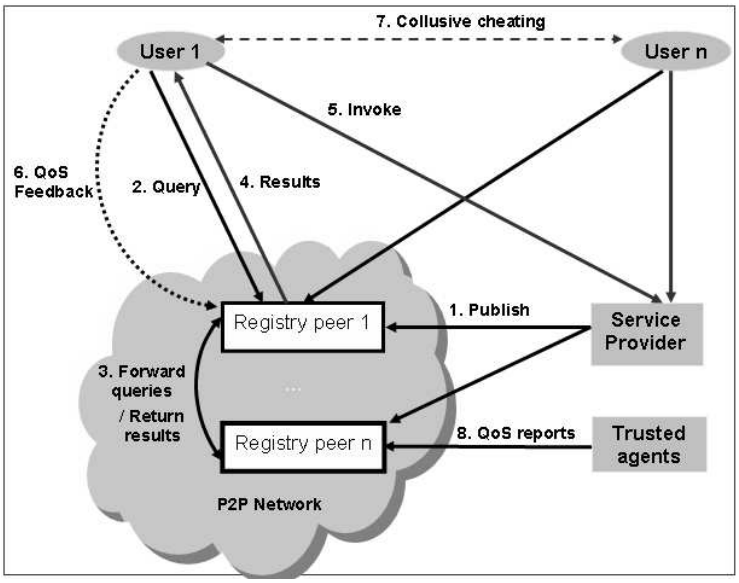

Fig. 1. Framework model

reports are combined with normal user reports to fine-tune the actual QoS characteristics of a service. In contrast to other models we do not depend on trusted agents but see them as an additional source of information and assume that only a small number of these agents exists as such services usually are costly to set up 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 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 QoS-based selection and ranking, based on QoS information provided in the service advertisements and QoS feedback data reported by the users. Providers are also able to query QoS of their own services and decide whether they should improve their services' performance or not.

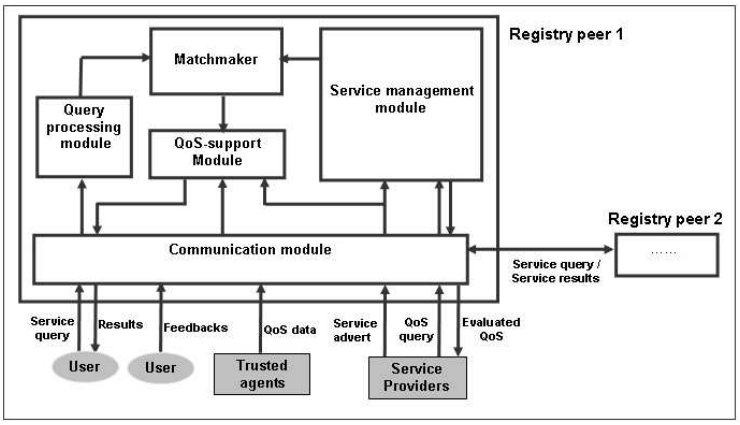

Fig. 2. Registry Peer Structure

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, expressed through ontology concepts as proposed by [13].
- optional QoS information in a specific QoS ontology, with the promised QoS for a specific operation or for the whole service.

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

- user's functional requirements in terms of service inputs, outputs, preconditions and effects, expressed by ontology 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 service description file using techniques suggested by WS-QoS [19] and Ran [23]. The ontology language we plan to use to describe service semantics and define the QoS ontology is WSMO [3], but other models, e.g., OWL-S would also be applicable. For experimental evaluations, we have developed a QoS ontology for the VISP use-case including the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, price, etc. Due to space limitations we cannot discuss this QoS model in any further detail here.

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 will be categorized into different *Concept Groups* based on their semantic similarity. Each group has a root concept defined as the one with the highest level in the ontology graph compared with the other member concepts. Without constraining general applicability we assume that all registries agree on one ontology of concepts. To uniquely represent a service query/advertisement independently of the order of service parameters, a total ordering of the concept groups is defined as follows:

Definition 1. *A concept group CG_x is considered as having higher order ($>$) than another group CG_y if one of these following conditions meets:*

1. *The level of CG_x in the ontology graph is higher than that of CG_y.*
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.*

A semantic service description, i.e., an advertisement or a query, is characterized by the concept groups that its input and output parameters belong to. [5] defines a mapping of ontological concepts onto numerical key values. 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 group's member, allowing us to quickly check the membership of any concept to that group [4]. For each input I_i (or output O_i) of a service, we find the concept group CG_i that it belongs to. *The characteristic vector* of this service is then represented by the ordered list of corresponding Bloom keys of all CG_is.

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.

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.

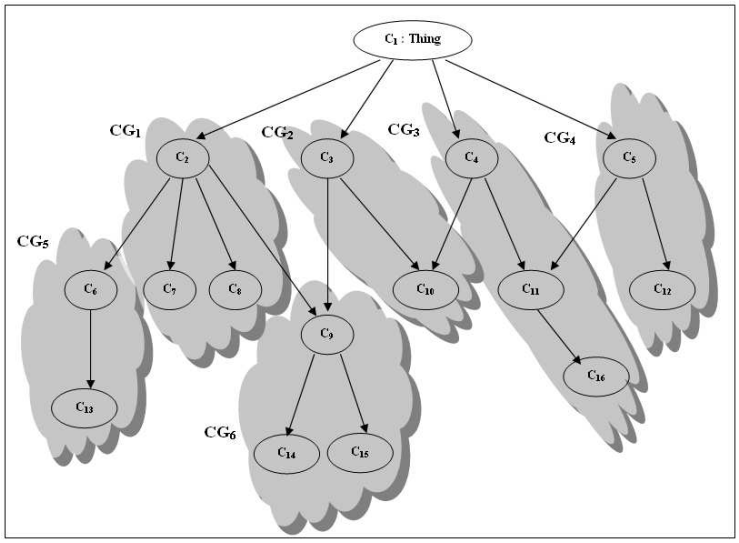

Fig. 3. Ontology graph partitioning

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. preconditions, effects, or service description keywords, 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.

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 = 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 search 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.

Eventually, the question of searching for a semantic service description becomes the problem of finding results for a multi-keyword query in the P2P network, which can be solved by using one of the two following approaches. The first one is simply concatenating all k_{ij}s together and then using this as the search key in the P-Grid network [2]. The second possibility is to deploy another type of peers in the network as *index peers* to keep identifiers of those registries managing keywords relating to various combination of k_{ij}s.

We 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 not only beneficial while searching for services with wildcard parameters but also advantageous for exchanging QoS reports and user information among neighboring registries later during the QoS predicting process.

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 [12]. However, their proposed method 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 a list L_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, $j = 1, \cdots, n$, and n is the number of services in this registry. The semantic similarity between two ontology concepts is computed based mainly on the distance between them in the ontology graph and the number of their common properties as defined by [11, 14].

A query for a service can be submitted to any registry peer and 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 having an approximate matching level d_i^j with rc_i, i.e., $d_i^j \approx d_i$, by browsing

the matching list 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 [12]. 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. Actually, we utilize another semantic matchmaking algorithm, e.g. [11], 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 most frequently accessed QoS attributes. The list of these attributes is initialized with popular QoS concepts, e.g., availability, reliability, execution-time, etc., and is updated periodically to capture changes in user demands. For other QoS attributes, the registry can derive them from the stored information of the published services and perform similar actions. With each QoS attribute q_j in this QoS matching table, we have a list $Lqos_j$ of records $\{S_{ij}, m_{ij}, promised_{ij}, predicted_{ij}\}$ where S_{ij} identifies a service, m_{ij} is the semantic similarity between q_j and the QoS attribute q_{ij} supported by S_{ij}, $promised_{ij}$ is the value of q_{ij} advertised by S_{ij}'s provider and $predicted_{ij}$ is the value of q_{ij} predicted by our QoS-based service selection and ranking engine, respectively. Apparently, we only store in $Lqos_j$ information of those S_{ij}s with m_{ij}s greater than a specific threshold. The values of $promised_{ij}$s and $predicted_{ij}$s should also be normalized regarding to service-specific and call-specific context information.

Given the list L of services with similar functionalities, the discovery engine performs the following QoS-based service selection and ranking algorithm:

Algorithm 1 QosSelectionRanking(ServiceList L, ServiceQuery Q)

1: Derive the list of QoS requirements in Q $L_q = [q_1, n_1, v_1], ..., [q_s, n_s, v_s]$
2: **for** each quality concept $q_j \in L_q$ **do**
3: **for** each service $S_i \in L$ **do**
4: Search the list L_{qos} of q_j for S_i;
5: **if** S_i is found **then**
6: $w_{ij} = weight(m_{ij})$; {m_{ij} is the semantic similarity between q_j and q_{ij}}
7: $PartialQosScore = w_{ij} \frac{predicted_{ij} - v_j}{v_j}$;
8: $QosScore[S_i] = QosScore[S_i] + \frac{n_j}{\sum n_j} PartialQosScore$;
9: **else**
10: Remove S_i from L;
11: **end if**
12: **end for**
13: **end for**
14: Return the list L sorted 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. Furthermore, we developed a probabilistic method to detect possible frauds when collecting user feedbacks with the assumption that these reports follow certain probabilistic distributions. Using trusted reports as reference values, we evaluate feedbacks of other users and apply a clustering algorithm to discover potential dishonest groups with different behaviors. Reports that fall out of a certain range of values are considered as incredible and will not be used in the predicting process.

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 un-ordered key sets where the keys are taken from a shared domain ontology. This indexing problem has not been addressed in the literature for structured overlay networks so far. 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 [11] and the service management module based on the UDDI standard. The existing implementation of the P-Grid system [2] is used as the basis for the communication module.

We also plan to extend our model such that registry peers are able to manipulate with different ontologies. Specifically, we will look into the problem of how to update and propagate ontologies more efficiently in the P2P registry network. Another enhancement would be to extend the classification criteria to include service preconditions, effects, and representative keywords of service textual descriptions. In addition, we are studying the possibility of developing and utilizing a caching mechanism to exploit the locality and frequency of service usages. One

more ambitious goal would be to add support for the composition of services in terms of QoS compatibility.

References

1. DIP Integrated project- Data, Information, and Process Integration with Semantic Web Services, http://dip.semanticweb.org/.
2. P-Grid: The Grid of Peers, http://www.p-grid.org/.
3. Web Service Modelling Ontologies, http://www.wmso.org/.
4. Space/Time Trade-offs in Hash Coding with Allowable Errors, Burton H. Bloom, Commun. ACM 13(7): p.p. 422-426, 1970.
5. Efficient Matchmaking And Directory Services, Ion Constantinescu and Boi Faltings, International Conference on Web Intelligence, Canada, 2003.
6. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, Abhijit Patil, Swapna Oundhakar, John Miller, Journal of Information Technology and Management, 2003.
7. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services, Mario Schlosser, Michael Sintek, Stefan Decker and Wolfgang Nejd, the Second IEEE International Conference on Peer-to-Peer Computing, 2002.
8. WSPDS: Web Services Peer-to-peer Discovery Service, Farnoush Banaei-Kashani, Ching-Chien Chen, and Cyrus Shahabi, International Symposium on Web Services and Applications, USA, 2004.
9. A Peer-to-Peer Approach to Web Service Discovery, Cristina Schmidt and Manish Parashar, World Wide Web Journal, Volume 7, Issue 2, June 2004.
10. A Peer-to-Peer Framework for Web Service Discovery with Ranking, Fatih Emekci, Ozgur D. Sahin, Divyakant Agrawal, Amr El Abbadi, Proceedings of IEEE International Conference on Web Services, USA, 2004.
11. Semantic Matching of Web Services Capabilities, Paolucci M., Kawamura T., Payne T., and Sycara K., Proceedings of the 1st International Semantic Web Conference, Italy, 2002.
12. Adding OWL-S to UDDI, implementation and throughput, Naveen Srinivasan, Massimo Paolucci, Katia Sycara, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, USA, 2004.
13. Adding Semantics to Web Services Standards, Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller, Proceedings of the International Conference on Web Services, 2003.
14. A Semantic Approach to Web Service Discovery, Abhijit Patil, Swapna Oundhakar, Ruoyan Zhang, Final Report on Semantic Web Course Spring 2002 (CSCI 8350), LSDIS Lab, Computer Science, University of Georgia.
15. Latest UDDI Version (3.0.2), UDDI Spec Technical Committee Draft, Dated 20041019, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.
16. Quality of Service in Service-Oriented Architectures, Glen Dobson, 2004, http://digs.sourceforge.net/papers/qos.html.
17. A Survey of Trust and Reputation Systems for Online Service Provision, Audun Jøsang, Roslan Ismail and Colin Boyd, Decision Support Systems, 2005 (to appear).
18. Possibilities for Managing Trust in P2P Networks, Zoran Despotovic and Karl Aberer, Technical Report, Swiss Federal Institute of Technology at Lausanne(EPFL), Switzerland, November 2, 2004.

19. WS-QoS - A Framework for QoS-aware Web Services, Andreas Gramm, Technical Report B-04-11, Institute of Computer Science, Freie Universität Berlin, Germany, July 2003.
20. UX - An Architecture Providing QoS-Aware and Federated Support for UDDI, Zhou Chen, Chia Liang-Tien, Bilhanan Silverajan, Lee Bu-Sung, Proceeding of the first International Conference on Web Services, 2003.
21. A DAML-Based Repository for QoS-Aware Semantic Web Service Selection, A. Soydan Bilgin and Munindar P. Singh, Proceedings of IEEE International Conference on Web Services, USA, 2004.
22. Reputation = f(User Ranking, Compliance, Verity), Sravanthi Kalepu, Shonali Krishnaswamy and Seng Wai Loke, Proceedings of IEEE International Conference on Web Services, USA, 2004.
23. A Model for Web Services Discovery with QoS, Ran S., ACM SIGecom Exchanges, Volume 4, Issue 1 Spring, pp. 1-10, 2003.
24. Efficient Access to Web Services, Ouzzani M., Bouguettaya A., IEEE Internet Computing, March/April, pp. 34-44, 2004.
25. A QoS Oriented Framework for Adaptive Management of Web Service based Workflows, Chintan Patel, Kaustubh Supekar, and Yugyung Lee, Database and Expert Systems 2003 conference, Prague, Czech Republic, 2003.
26. Reputation and Endorsement for Web Services, E. Michael Maximilien and Munindar P. Singh, SIGEcom Exchanges, 3(1):24-31, Winter 2002, ACM Special Interest Group on E-Commerce.
27. QoS computation and policing in dynamic web service selection, Yutu Liu, Anne Ngu, and Liangzhao Zheng, Proceedings of the WWW 2004, 2004.
28. Selecting the Best Web Service, Julian Day and Ralph Deters, the 14th Annual IBM Centers for Advanced Studies Conference, 2004.
29. Statistical Fraud Detection: A Review, Richard J. Bolton and David J. Hand, Statistical Science, 17(3), 235-255, January 2002.
30. Managing Trust in a Peer-2-Peer Information System, Karl Aberer and Zoran Despotovic, 10th International Conference on Information and Knowledge Management (ACM CIKM), 2001.
31. The EigenTrust Algorithm for Reputation Management in P2P Networks, Sepandar D. Kamvar, Mario T. Schlosser and Hector Garcia Molina, Proceedings of the Twelfth International World Wide Web Conference, 2003.
32. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities, Li Xiong, Ling Liu, IEEE Transaction on Knowledge and Data Engineering, p.p 843-857, 2004.
33. Filtering Out Unfair Ratings in Bayesian Reputation Systems, Andrew Whitby, Audun Jsang and Jadwiga Indulska, Proceedings of the Workshop on Trust in Agent Societies, USA, 2004.
34. Probabilistic User Behavior Models, Eren Manavoglu, Dmitry Pavlov and C. Lee Giles, Third IEEE International Conference on Data Mining, Melbourne, Florida, 2003.
35. Choosing Reputable Servents in a P2P Network, Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi and Pierangela Samarati, Proceedings of WWW 2002 Conference, USA, 2002.