A Peer-to-Peer Approach to Semantic Web Services Discovery*

Yong Li, Sen Su, and Fangchun Yang

State Key Lab. of Networking and Switching,
Beijing University of Posts and Telecommunications
liyong.bupt@gmail.com, {susen, fcyang}@bupt.edu.cn

Abstract. Web Services with distributed and dynamic characteristics need efficient and decentralized discovery infrastructure supporting the semantic descriptions and discovery. In this paper, a novel peer-to-peer indexing system with related P2P registries is proposed to support the completely decentralized discovery capabilities. In the presented system, with the ontology encoding scheme, the semantic service description is distributed into a distributed trie index on the structured P2P network to allow requesters to lookup services with semantic requirements. Finally, experimental result shows that the presented system is efficient and scalable.

1 Introduction

Web Services are emerging the most popular paradigm for distributed computing. Using Web Services technologies the enterprises can interact with each other dynamically. A typical Web Service architecture consists of three entities: service providers that publish Web Services, service brokers that maintain support their discovery, and service requesters that invocate Web Services.

The growing number of Web Services demands for a scalable, flexible and reliable solution to discovery the most appropriate services for the requesters. The mechanisms of service discovery include centralized registry and decentralized approach. Much of work on Web Services discovery is based on the centralized registries, like UDDI [1] or DAML-S matchmaker [2]. However as the number of Web Services grows and become more dynamic, the centralized registries, which lead to a single point failure and performance bottleneck, quickly become impractical.

In order to avoid the disadvantages of the centralized systems, a number of decentralized solutions based on P2P technologies have been proposed. Some systems build on P2P network use ontologies to publish and discover the web services descriptions. The systems depend on classification or metadata routing [4, 5] can offer rich mechanism of query services. However, the unstructured P2P network limits the

* This work is supported by the National Basic Research and Development Program (973 program) of China under Grant No. 2003CB314806; the National Natural Science Foundation project of China under Grant No.90204007; the program for Changjiang Scholars and Innovative Research Team in University (PCSIRT);. National Natural Science Funds for Distinguished Young Scholar(No.60125101).

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part IV, LNCS 3994, pp. 73 – 80, 2006.

scalability of these approach. The structured P2P networks based on Distributed Hash Table (DHT) [3] are extremely scalable and lookups can be resolved in logn overlay routing hops for a network of size n nodes. However, DHT overlays support only "exact match" lookups and encounter difficulties in complex queries. Although some DHT-based P2P systems [6, 7] are proposed to enhance capabilities of query, there still lack of open and efficiency solutions for the web services with semantic advertisement and discovery.

In this paper, we propose a scalable system that support semantic service discovery by publishing advertisement of semantic Web services on a structured overlay network. The Web services can be described in semantic method, according to existing standards (e.g. OWL-S [14]), and can be characterized by a set of keywords. We use these keywords to index the Web services descriptions, and store the index at peers in the P2P systems using a DHT approach. In order to support semantic queries, we use multiple ordered keywords sets taken from domain ontologies as index terms for semantic web service descriptions.

The presented system uses the semantic overlay on top of DHT to manage service advertisements. We deploy a distributed trie index on the semantic overlay to support semantic services matchmaking containing subsumes and overlaps. The approach makes it possible to quickly identify the peers containing most likely matched services according to user requests by the combination of the ontology numerical naming scheme.

The rest of this paper is structured as follows. Section 2 compares the presented system to related work. Section 3 introduces our model of Web Services discovery. Section 4 describes the architecture and operation of the presented distributed indexing system. Section 5 shows an experimental evaluation of the system. Last section concludes the paper and presents future work.

2 Related Work

Currents research to Web service discovery can be broadly classified into two categories: centralized and decentralized approaches.. The centralized approaches include UDDI [1], where central registries are used to store Web Service descriptions. Most of the decentralized approaches using p2p structure and combine with ontology to discover the web services. [9, 10] are similar to our method as they are built on top of structured P2P systems. [9] Describe Web Services by a set of keywords and then map the multi-dimensional index to a DHT via linearization. Some service discovery systems adopt ontology-based approach to improve efficiency. [8, 11] make use of ontology to organize web service discovery registries and addresses scalability of the discovery process.

3 System Architecture

Fig 1(a) shows the architecture of our system. It consists of three layers:

The DHT layer is designed as a common communication layer, on which higher level service discovery mechanism can be built. The DHT layer ensures a scalable

management and routing for overlay network. The semantic overlay layer is in turn built on top of a scalable distributed hash table, and formed by the semantic description of service.

Distributed trie built on top of the semantic overlay can perform two operations, register a service description or issue a query to search for service. The main task of the distributed trie layer is to determine where to register a service description and where to send a service query. Queries are directly sent to the trie root for resolution according to the keys in the queries, and searching the whole network is not needed.

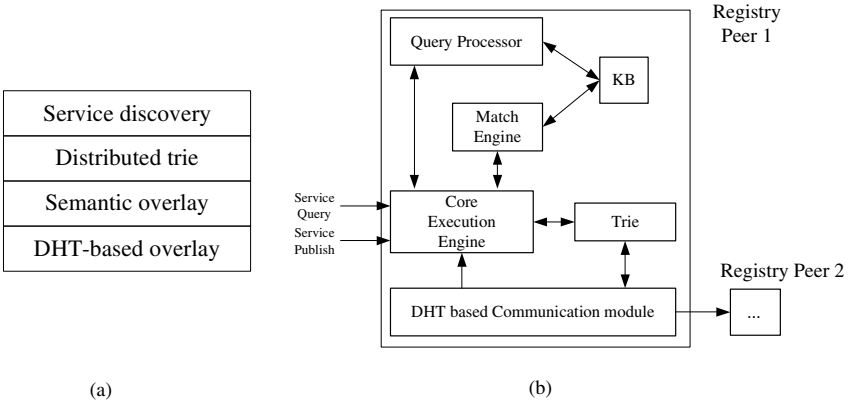

Fig. 1. Architecture of the system. (a) Layered model of the discovery system and (b) structure of registry node.

Fig .1(b) shows the internal structure of registry node. The core execution engine module connects the other internal components and interacts with users, i.e., publish services, submit queries and receive results. The DHT-based communication module takes charge of underlying P2P network and deals with the connection to other registry peers. The query processor transforms service semantic description into numeric keys that our system can be handled. The numeric key set of service is then sent to the root of distributed trie, and the trie module perform the service publish and query on the distributed trie. After the distributed trie return the service advertisements to the node issuing service query, the match engine with knowledge base will complete ontology-based semantic matching between service query request and service advertisements.

4 Service Description, Registration and Discovery

4.1 Mapping the Semantic Service Description to Keys

In our system, a semantic service description, i.e., a service advertisement, a service request, will be associated with the *characteristic vector*, a set of numeric keys of ontological concepts for the description of the service. Using characteristic vector, service advertisements are assigned to registry peers. Similarly, requesters can discover registries through the vector of the service requests.

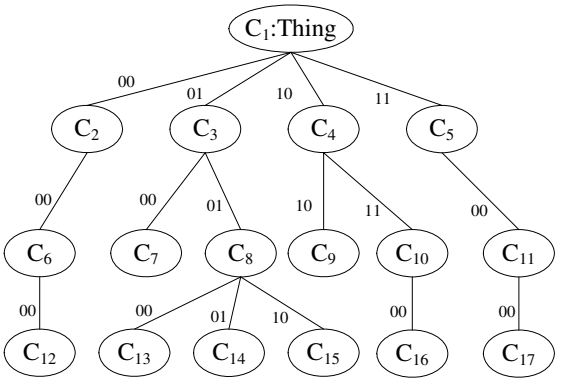

Fig. 2. Ontology numeric naming scheme

The ontological concept, as the elements of characteristic vector, will be encoded to numeric keys. The detailed process is described in the following part.

We model the ontology graph as a multibit-trie where each ontological concept is a node. We use the maximal out degree of the ontological graph to the fixed-stride of the trie. For each node of the trie, we label each branch in a binary string from zero to maximal out degree of ontological graph. We define the key of the ontological concept as a string composed by the branch labels from root to the node representing the concept. This encoding scheme make the code of concepts have the prefix property, which represent the hierarchical relationship among ontological concepts.

The elements of characteristic vector sort in descending order of concept C_i. The descending order of C_is is defined like breadth-first search of tree-like trie structure. A concept C_i have higher order than another concept C_j, if (1) the level of C_i is higher than the level of C_j , (2) both C_i and C_j have the same level and C_i is in the left of C_j in the ontology graph.

An example of ontology numeric encoding scheme is illustrated in Fig. 2, where C_i is an ontological concept. In this graph, the maximal out-degree of nodes is 4, so, from left to right, the branches of C_8 are represented by 00, 01, and 10, and the branches of C_1 are represented by 00, 01, 10 and 11. k_8, the encoding of C_8, is "0101" and k_{15} is "010110". Apparently, according to the encoding of the concepts, we can notice that C_{15} is the child concepts of C_8.

4.2 Publishing Service Advertisements to Peer Registries

Using the numeric encoding of concept as the key of the DHT function, each concept in ontology graph is published on a peer of the object overlay. In this method, the nodes with semantically related concepts form a semantic overlay

We deploy a distributed trie index on the semantic overlay. A distributed trie index is a tree-like index that supports complexity queries on key strings. A trie can be viewed as an m-ary tree where the keys are arranged on extra leaf nodes. Keys with the same k prefix share the same path of k indexing nodes from the root to leaves [13]. Searching a key on a trie index starts from the root and follows the node that meet the query along a trie path until arriving a leaf node.

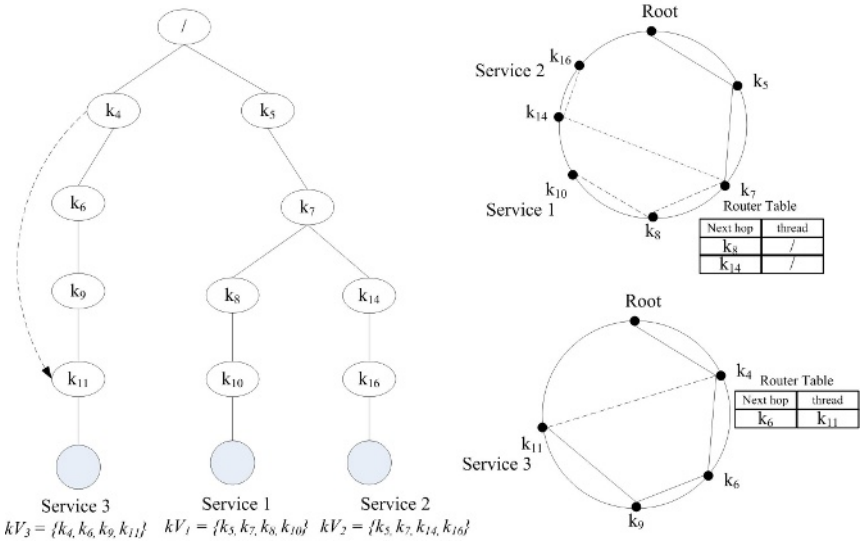

Fig. 3. Service publishing and discovery on distributed trie

We use the characteristic vector of service as the trie path. Publishing a web services is a searching and publishing process of the characteristic vector. As Fig. 3 shows, the method to build a distributed trie and publish the service to overlay works is:

1. For a characteristic vector $CV = \{k_1, k_2 \ldots k_n\}$ arriving at root node, if the first key k_1 was not in the router table, insert k_1 , forward service description to the node k_1.
2. For node k_i in CV, when service description arrived, the router table of k_i will be checked. If there is not exists the next node k_{i+1} in router table, insert it; and service description will be forwarded to k_{i+1}.
3. According to 2, a trie path will be formed by the characteristic vector $\{k_1, k_2 \ldots k_n\}$, if there exists a partly trie path tp_1 $(k_1\ k_2 \ldots k_m)$, build the rest part tp_2 $\{k_{m+1}\ k_2 \ldots k_n\}$ of the characteristic vector after the tp_1.
4. Finally, service description arrives at the last node k_m and is stored in this node.

Using distributed trie index, the problem of searching for the registries most likely to store a matched service becomes the problem of finding peers along with trie path. This method of service registration will help us quickly look up the "right" registry peer with a service request in the following discovery process.

The efficiency of a trie index, mainly search hops, is proportional to the average depth of the trie and length of keys. A full trie index has much longer search paths and many hops. Besides the full trie, there are some types of trie to reduce search hops and increase efficiency: pruned trie [13], Patricia trie [13] and compressed trie [12]. However, these trie indexes need to move and adjust existing services from nodes to the others when publishing new services; so, they are not suitable for the distributed environment. We devise a threaded trie index to reduce average search hops and avoid moving existing service advertisement. As the Fig.3 shows, from the

node published service, if there are a chain only consisted of 1-degree nodes, the first node that has only one successor will set up a trie pointer directly link to the node published service. When a service query arrived at a node, if the node has found the next hop with a point to service registration, it sends the service query to the node directly. In this method, we reduce search hops to increase discovery efficiency.

4.3 Service Discovery on Distributed Trie

Basic queries and semantic matching can be supported by distributed trie on semantic overlay. We first need to introduce the definition of the matchmaking degree described in [2]. The outcome of matchmaking between service request Q and service advertisement S could be one of the types below:

- **Exact** - If advertisement S and request Q are equivalent, we call the match Exact, formally, $S \equiv Q\ \psi$
- **PlugIn** - if S could always be used for Q. S is a plug-in match for Q, formally, $Q \subseteq S$.
- **Subsume** - if S is more general than Q, we call the match subsume. formally, $S \subseteq Q$

In order to query a service in the distributed discovery system, a requester needs to send query to a participating peer, called *introducer*. The introducer first checks local service registrations. If presented, the discovery process terminates. Otherwise, query request will be forwarded to the root of distributed trie and begin a distributed discovery process. To search a service description, we convert the service description to its characteristic vector as mention in 4.1, and then look for it on the trie starting at the root. When the service query arrive at node registering concept k_i, if the next node k_{i+1} is in the router table, the service query can be forwarded to the k_{i+1} for "*Exact*" query. Meanwhile, the service query will be forwarded to the nodes are found to satisfy "*Subsume"* or "*PlugIn"* with k_{i+1} by browsing the router table. Whenever the service requirement reaches the node of registered service, the node will return the service description to service introducer.

Regarding Fig. 4, suppose that we have two services advertisements published in distributed trie: S_1 described with C_5, C_7, C_8, and C_{10} and S_2 described with C_5, C_7, C_{14}, and C_{16}. The characteristic vector CV_1 of S_1 will be $\{k_5, k_7, k_8, k_{10}\}$ and CV_2 will be $\{k_5, k_7, k_{14}, k_{16}\}$. According to our way of distributed service registration, to publish S_1, a *trie path* $<k_5, k_7, k_8, k_{10}>$ *is generated and* S_1 *is registered to the last node* k_{10}, *likewise,* S_2 is attached in k_{16}. In case the characteristic vector of service query is $\{k_5, k_7, k_{14}, k_{16}\}$ and the match requirement is "*Subsume"*, after the service query arrive at node k_7, according to matching requirement, the node will choose the next node by browsing the router table and send the service query to the selected nodes: k_8 and k_{14}. The nodes k_8 and k_{14} will return the service description of S_1 and S_2 to the requester.

5 Experimental Evaluation

Accessing a concept encoding requires $O(log_2N)$ hops through the DHT function of Chord overlay where N is the number of physical nodes. Locating a characteristic vector using distributed trie takes $O(W*log_2N)$ hops on Chord, where W is the length of a characteristic vector.

We evaluate the performance of our system by measuring average search hops. Two cases are tested: full trie and threaded trie. To measure average search hops, we simulate distributed trie on a chord simulator implemented in java [15]. We use a sample ontology described in [14] and generate numeric encoding for each concept. We generate randomly 1000 characteristic vectors which are uniform distribution. The average characteristic vector length is 6.2. We measure the average search hops in the network with different numbers of nodes. Fig. 4 shows the threaded trie outperforms full trie approach. The average search hops is low and also increases gracefully with increasing number of nodes. For example, the average search hops is 16.3 in a 50 node system, whereas it takes 22.1 when the node number increases to 500. The actual number of nodes in our system can be much more than the number shown in the graph. Thus we conclude that out system is both scalable and efficient in terms of discovery. Compared with a centralized service discovery system, our system is more scalable. And it also supports semantic match requirement of discovery operations at the increase of a little latency cost.

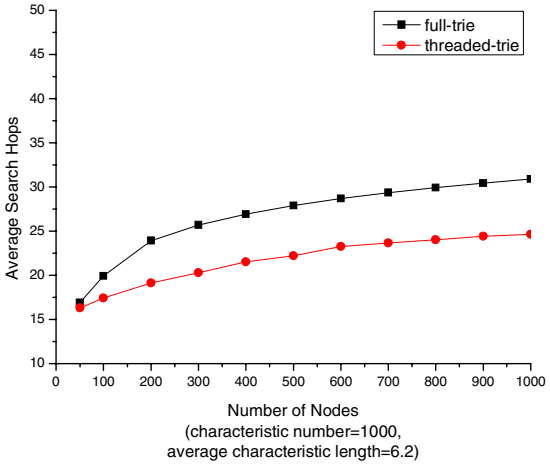

Fig. 4. Average search hops on the distributed trie with different number of nodes

6 Conclusion

In this paper, we propose a semantic service discovery system based on P2P overlay network. The present system can support semantic matching requirements of services on the structured overlay network through a distributed trie index and the ontological concept encoding scheme. We present how a distributed trie is deployed on structured P2P system to support service query with semantic requirement. An experimental evaluation shows the good scalability of the system. In the future work, we plan to improve the usability of our system through supporting both the keyword-based and the ontology-based service discovery requirements.

References

1. http://www.uddi.org/.UDDI Version 3.0, Publish Specification.
2. M.Paolucci, T.Kawamura, T.R.Payne, and K.sycara. Semantic matchmaking of web services capabilities. ISWC2002,2002
3. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. Proceedings of ACM SIGCOMM`01, San Diego, September 2001.
4. 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.
5. W. Nejdl, M. Wolpers, W. Siberski, A. Loser, I. Bruckhorst, M. Schlosser, and C. Schmitz: Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks. In Proceedings of the Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, May 2003.
6. Harren, JM Hellerstein, R Huebsch, BT Loo: Complex Queries in DHT-based Peer-to-Peer Networks IPTPS, 2002.
7. H.T. Shen, Y. Shu, and B. Yu, "Efficient Semantic-Based Content Search in P2P Network," IEEE Trans. Knowledge and Data Eng.,vol. 16, no. 7, pp. 813-826, Aug. 2004.
8. M. Paolucci, K. P. Sycara, T. Nishimura, and N. Srinivasan. Using daml-s for p2p discovery. In Proceedings of the International Conference on Web Services, pages 203–207, 2003.
9. Schmidt, C. and Parashar, M. A peer-to-peer approach to Web service discovery, World Wide Web, 7 (2) (2004) 211-229.
10. L.- H. Vu, M. Hauswirth and K. Aberer: Towards P2P-based Semantic Web Service Discovery with QoS Support, Proceeding of Workshop on Business Processes and Services (BPS), Nancy, France, 2005
11. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A scalable and ontology-based P2P infrastructure for semantic web services. In Proceedings of the Second International Conference on Peer-to-Peer Computing, pages 104–111, 2002.
12. K.Maly. Compressed tries. Communications of the ACM, 19(7):409-15,1976.
13. D.E.Knuth. Sorting and Searching, volume 3 of The Art of Computer Programming. Addison-Wesley,Reading, MA, 1973
14. The OWL-S Service Coallition: OWL-S: Semantic Markup for Web Services, version 0.1. http://www.daml.org/services/owl-s/1.0/owl-s.pdf.
15. Pedro García, Carles Pairot. PlanetSim: A New Overlay Network Simulation Framework. Workshop on Software Engineering and Middleware (SEM 2004).ï