Performance Improvement in Web Services Discovery for Mobile Networks

Li-Der Chou, Po-Chia Tseng and Jyun-Yan Yang
Department of Computer Science and Information Engineering, National Central University
No. 300, Jhongda Rd., Jhongli City, Taoyuan County 320, Taiwan (R.O.C.)
E-mail: cld@csie.ncu.edu.tw

Abstract- *Service-Oriented Architecture (SOA) provides a scalable and interoperable paradigm for organizing distributed resources. In general, the SOA is implemented by Web services, and discovery service is the fundamental process of services engagement in Web services. However, a centralized architecture for Web services discovery, such as Universal Description and Discovery Integration (UDDI), goes through the single node of failure and scalability problem. Otherwise, the existing peer-to-peer architectures, such as Gnutella-based or Distributed Hash Table-based, were suffered from larger delay due to multi hops delivery in mobile networks especially. In this paper, an improving Web services discovery in mobile networks is proposed, which is a decentralized and adaptive service discovery with enhancement of matching capability. The architecture adopts a distributed index system with support of Pastry overlay networks for services discovery. The keyword query for semantic search in peer-to-peer network is supported in the proposed architecture as well. Furthermore, the routing procedure and leaf set of Pastry is adopted to reduce delay in mobile networks. The simulation show that our proposed solution is scalable and improved the performance up to 67.85% compare to Chord-based solutions in mobile networks.*

Keywords: Web services discovery, Peer-to-Peer, Mobile Networks, Delay.

1. Introduction

In the Web services architecture (WSA), there are four steps in service engagement: “Parties become known to each other”, “Agree some semantics and Web Services Description (WSD)”, “Input WSD and Semantics to agent”, and “Interact”. “Parties become known to each other” is also called Web services discovery and required as a primitive functionality for service requesters to locate and match services. The definition of “discovery” is “the act of locating a machine-processable description of a Web service that may have been previously unknown and that meets certain functional criteria” given by WS Glossary.

The Web service discovery can be classified as centralized and decentralized. Centralized architecture, such as Universal Description and Discovery Integration (UDDI), has only one centralized registry or index to collect the address of service endpoints, and centralized architecture is appropriate in networks where the locations of nodes do not change frequently [3]. On the contrary, decentralized architecture is composing of many broker agents distributed in Internet, and adaptable in mobile networks in which proximity limits the need to propagate requests [5].

We elaborate a Pastry-based Web services discovery adapts to larger scale networks and declines average delay in mobile networks [6][7]. For the generic services lookup and advertisement functions of the architecture are implemented in a simulator environment. The architecture adopts a distributed index system with support of Pastry overlay networks for services discovery. The keyword query for semantic search in peer-to-peer network is supported in the proposed architecture as well. Furthermore, the routing procedure and leaf set of Pastry is adopted to reduce delay in mobile networks. The simulation show that our proposed solution is scalable and improved the performance up to 67.85% compare to Chord-based solutions in mobile networks.

The rest of this paper is organized as follows. In section 2, related work to Web services discovery is discussed. Section 3 proposes assumptions and the architecture of Web services discovery. Simulations and discussions are conducted in section 4. Finally, Section 5 gives the conclusions.

2. Related Work

In this section, decentralized P2P-based Web services discovery will be discussed. The P2P solutions are addressed and compared as follow.

The Web Services Peer-to-peer Discovery Service (WSPDS) [1] is a decentralized design for the Web Services discovery. WSPDS depicts an unstructured peer-to-peer network of WSPDS servants. The servants of WSPDS collaborate to resolve discovery queries raised by their peers. Each servant is composed of two engines, communication engine and local query engine. The communication engine provides the interface to user and also represents the servant in the peer-to-peer network of servants. Local query engine receives queries from the communication engine, queries the local site for matching services, and sends responses to communication engine.

The SPiDeR built a super-peers overlay on top of Chord to select Web services efficiently [2][8][9]. Chord is a lookup service that implements a Distributed Hah Table (DHT). It uses a *m*-bit identifier ring, which is $[0, 2^m\text{-}1)$ for routing and locating objects and peers. Both objects and peers are assigned *m*-bit keys through a uniform hash function, such as SHA-1. An object is stored at the peer following it on the ring, which is called its successor.

SPiDeR allows distributed Web service discovery over a P2P system based on Chord. It provides several

This research was sponsored in part by National Science Council under grants NSC 95-2627-E-008-002 and NSC 96-2628-E-008-004

techniques to increase the accuracy of service discovery including functional matching (what a service does), behavioral matching (how a service performs), semantic matching (the underlying semantics of a service) and ontological matching (how a service relates to other services). Each of these provides a different metric to measure the relevance among different services, and the goal of SPiDeR is to achieve above matching mechanisms. However, the issues of performance are not discussed in SPiDeR.

The pService Web services discovery system is similar to other P2P discovery systems which use keywords as indexing keys to complete resources lookup [3][4]. That means metadata in service descriptions should be extracted as keywords to index services. To support complex search and semantic query, Web Service Description Language (WSDL) is schemed as service description language. In the research, the most important description is hashed into the Chord overlay, and other service descriptions are hashed into Skip Graph.

The message routing algorithm is based on the Chord. Thus, the number of average hops for Web service discovery inherits from Chord. It shows that the number of average hops is log$N$ in the N-node network.

3. System Architecture

Performance Improvement in Web Services Discovery (PIWSD) for mobile networks, there are three components proposed to realize the peer-to-peer Web services discovery, which are service advertisement function, service lookup function, and routing algorithm, respectively.

3.1. Service Advertisement

This function is embedded in brokers. It defines the procedure of how provider agents register their services into given discovery service agents, and so called “service advertisement”. In order to maintain the interoperability, it is necessary to formalize the description of services, such as WSDL proposed by W3C. Besides, we use a common Semantic Markup for Web Services (OWL-S) [10] to advertise the services from service requester to provider.

The WSDL contains the information about “how to use the service”, and the OWL-S contains the information about “what the service is”. The PIWSD extract keywords for a service from its OWL-S file, and use the routing algorithm to route the keywords stored in the corresponding nodes. The procedure of service advertisement differs from the original Web Services Architecture (WSA), so we would describe the process in Figure 1. There are three components in the service advertisement:

1) Provider Agent: the agent is capable of and empowered to perform the actions associated with a service on behalf of its owner that is providing the Web service.

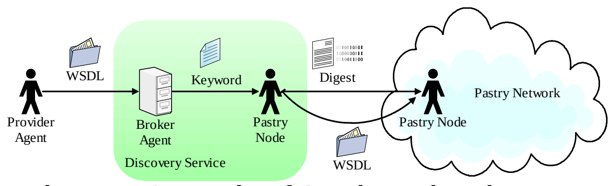

Figure 1. Scenario of Service Advertisement

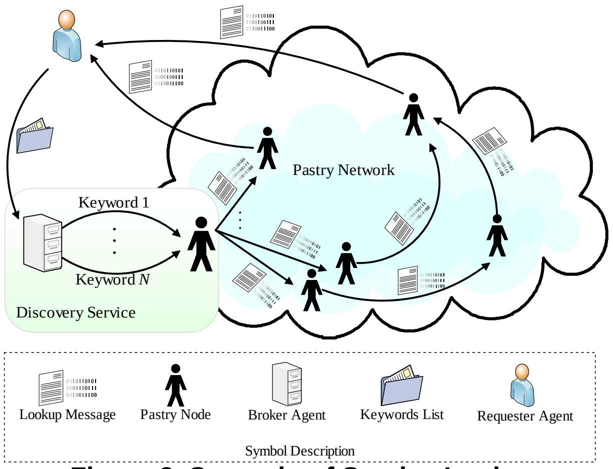

Figure 2. Scenario of Service Lookup

2) Discovery Service: it is a service that enables broker agents to extract keywords in WSDL file and also play a role as a Pastry node to spread WSDL files into Pastry network.

3) Pastry Node: this node efficiently routes the message to the other nodes with a unique numeric identifier *nodeId*.

3.2. Service Lookup

This function is to let requester agent gets service endpoints from discovery services, and these service endpoints meet the criteria requested by requester agent. Also, this function would call the routing algorithm to locate where the keyword is. In the service lookup, there are three components including requester agent, discovery service and pastry node shown as Figure 2. Requester agent is a software agent that wishes to interact with a provider agent in order to request that a task be performed on behalf of its owner. Discovery service and pastry node are defined well that mentioned at section 3.1.

While a service requester wants to invoke a service, the requester agent will send the keyword list of requested service to the broker agent. Then the broker agent will extract keywords from the criteria file and deliver keywords to the pastry node in order to search the services related to those keywords. Thus, the pastry node will wrap the keywords as a lookup message and route the lookup message to the corresponding pastry node until the pastry node who keeps the service information related those keywords are founded. The target pastry node will return the WSDL file to the requester agent.

3.3. Routing algorithm

The function is use to locate where the keyword is, whenever a message with key D arrives at a node with *nodeId A*. This algorithm is based on Pastry’s routing algorithm, which can find a given keyword within $\lceil \log_{2b} N \rceil$ steps for N nodes pastry networks, b is a configuration parameter with typical value 16.

When Pastry node A receives a lookup message has a keyword D, the node checks whether the D falls within the range of *nodeIds* covered by its leaf set. If so, the message is forwarded directly to the destination node, namely the node in the leaf set whose *nodeId* is closest to the key (possibly the present node).

If the D is not covered by the leaf set, then the routing table is used and the message is forwarded to a node that shares a common prefix with the D by at least one more digit. In certain cases, it is possible that the appropriate entry in the routing table is empty or the associated node is not reachable, in which case the message is forwarded to a node that shares a prefix *nodeId* with the D at least as long as the local node, and is numerically closer to the D than the present node's id.

Such a node must be in the leaf set unless the message has already arrived at the node with numerically closest *nodeId*. And, unless $\lfloor |L|/2 \rfloor$ adjacent nodes in the leaf set have failed simultaneously, at least one of those nodes must be live.

4. Simulations and Discussions

In this section, experiments have been conducted in order to evaluate the effectiveness and efficiency of Pastry-based Web Service discovery (PIWSD). These experiments were performed in a P2P simulation system called PeerSim. There are agents that would generate the traffic of lookup message, and the peer-to-peer network would responsible for messages routing. The hash function is SHA-1 and the length of *nodeId* is 160 bits in our simulations.

For the comparison of previous work, we also implement a Chord-based discovery to compare the system performance. In the simulation result, it reveals that our approach is cost lower than the Chord-based approached in numbers of hop counts. It is approximately reduced the number of hops to 50%.

This experiment on average hops and delay aims to compare this approach to previous research based on Chord. There are 1024 nodes in this experiment. And the traffic generator will generate the traffic every 100 millisecond. There is no turbulence in this simulation. The total simulation time is 3×10^5 ms and the related parameters to Pastry overlay are: the base is 2 and size of leaf set is 32. And the end-to-end delay of transport layer is random distribution between 500 milliseconds to 900 milliseconds.

According to the simulation result, we can see that average service discovery delay of PIWSD is 67.8% of Chord-based Web service discovery in Figure 3. This is due to the complexity of routing algorithm between Chord-based and PIWSD. Although the complexity of routing table of PIWSD is more complex than Chord-based solutions. The improvement of average number of hops still causes the improvement of average delay.

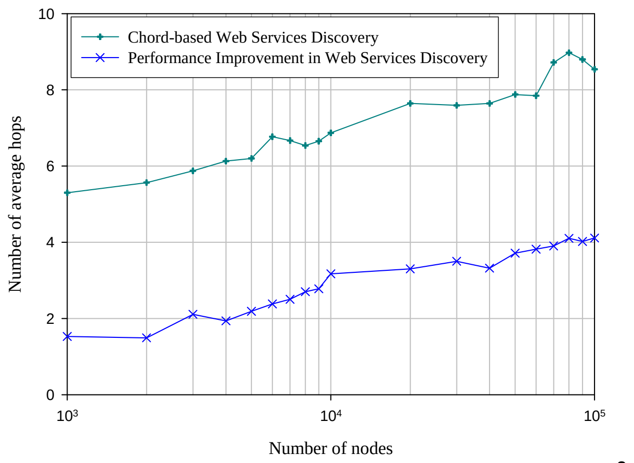

Figure 3. Number of Average Hops Between 10^3 to 10^5 Nodes

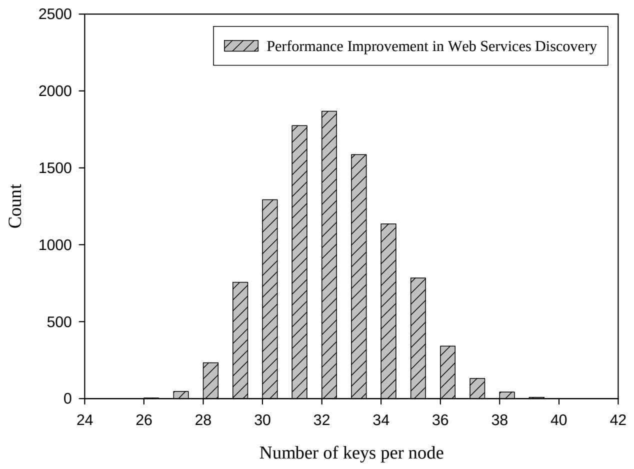

Figure 4. The Counts of Number of Keys Per Node in PIWSD. The Total Number of Key is 10^4

The goal of experiment on capacity distribution aims to examine the capacity distribution on PIWSD and Chord-based Web Services Discovery. The distribution will reveal that the capacity usage for storing the number of keys. Given a N total keys needed to be stored, the number of keys per node stored is depend on N. If the number of keys per node needed to be stored is fewer, the cost of capacity is less.

In Figure 4, it shows that the counts of the number of keys per node stored in Performance Improvement in Web Services Discovery (PIWSD). Also, is distributed by hash function in the routing table, whose *nodeID* is in base 4. In Pastry-based routing table, the size of routing table is approximately $\lceil \log_4 N \times (4-1) \rceil$. So, the counts of the number of keys per node in PIWSD will more than Chord-based. The result shows that the average number of keys per node in PIWSD is 32.0725, and 13.6243 in Chord-based. It means our proposed PIWSD costs numbers of keys more than Chord-based up to 18.4482 keys. Also, the scalability is also concerned.

5. Conclusion and Future Work

The P2P overlay is built on top of Pastry in order to distribute service endpoints between peers. And three necessary components of Web service discovery in P2P networks are defined as follow: service advertisement, service lookup and routing algorithm. Compare to Chord-based solutions, the experimental result real that the proposed PIWSD improve the delay up to 67.8%. Due to decreased hops of look up messages, the delay is reduced compare to Chord-based and the hops while searching a given key is bounded in log *N*, where *N* is the number of keys. Also, the effect on load distribution is conducted to show that our proposed PIWSD will avoid the single node of failure problem. Besides, in a dynamic environment with node join or leave, the experiment is conducted to show the cost of a node join and leave. For consideration of cost for capacity per node, the experiment on capacity distribution reveals that our proposed PIWSD cost more capacity to store routing information than Chord-based.

In the future, the implementation of the proposed PIWSD is taken into consideration to various applications, such as mobile social networks or mobile Web services composition. Also, we can deploy the proposed functionality to provide public use in the Internet..

References

[1] F. B. Kashani, C. C. Chen and C. Shahabi, "WSPDS: Web Services Peer-to-peer Discovery Service," *Proceedings of International Symposium on Web Services and Applications*, Nevada, USA, pp. 733-743, Jun. 2004.

[2] O. D. Sahin, C. E. Gerede, D. Agrawal, A. E. Abbadi, O. H. Ibarra and J. W. Su, "SPiDeR: P2P-Based Web Service Discovery," *Proceedings of International Conference on Service Oriented Computing*, Amsterdam, The Netherlands , pp. 157-169, Oct. 2005.

[3] W.F. Lv and J.J. Yu, "pService: Peer-to-Peer based Web Services Discovery and Matching," *Proceedings of ICSNC 2007*, French Riviera, France, pp. 54-54, Aug. 2007.

[4] G. Zhou, J. J. Yu, R. Chen and H. Zhang, "Scalable Web Service Discovery on P2P Overlay Network," *Proceedings of IEEE International Conference on Services Computing*, Utah, USA, pp. 122-129, Jul. 2007.

[5] U. Srivastava1, K. Munagala, J. Widom and R. Motwani, "Query Optimization over Web Services," *Proceedings of the 32nd international conference on VLDB*, Seoul, Korea, vol. 32, Sep. 2006.

[6] S. Kona, A. Bansal, G. Gupta and T. D. Hite, "Semantics-based Efficient Web Service Discovery and Composition," *Proceedings of ISWC 2004*, Hiroshima, Japan, Nov. 2004.

[7] D. Wu, T. Tian and K. W. Ng, "An Analytical Study on Optimizing the Lookup Performance of Distributed Hash Table Systems under Churn," *Transaction on Concurrency and Computation*, vol. 19, pp.543-569, Jan. 2007.

[8] E. Ayorak and A. B. Bener, "Super Peer Web Service Discovery Architecture," *Proceedings of ICDE 2007*, pp. 1360-1364, Apr. 2007.

[9] T. Yu, Y. Zhang and K. J. Lin, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints," ACM Transactions on Web, vol. 1, iss. 1, May 2007.

[10] N. Srinivasan, M. Paolucci and K. Sycaral, "CODE: A Development Environment for OWL-S Web services," Proceedings of ISWC 2004, Hiroshima, Japan, Nov. 2004.