ABSDM: Agent Based Service Discovery Mechanism in Internet[1]

Shijian Li, Congfu Xu, Zhaohui Wu, Yunhe Pan, and Xuelan Li

Zhejiang University Artificial Intelligence Institute, Hangzhou, 310027
Jian1231@yahoo.com, {Xucong, wzh}@cs.zju.edu.cn,
panyh@sun.zju.edu.cn, gracelxl@sina.com

Abstract. To improve popular services discovery mechanism (UDDI mainly), we propose an agent-based services discovery mechanism. In this mechanism, services information is stored in distributed servers that are regarded as independent agents. These servers are joined into a tree structure; and then, a recursive algorithm was used to distribute searching request over the whole tree. Based on this condition, searching request could be rapidly parallel dealt with.

1 Introduction

Nowadays, UDDI (Universal Description, Discovery, and Integration) was used mostly for services discovery process [1]. But it has some disadvantages as follows [2]: (1) The condition that tremendous WSDL documents are stored on centralized UBR (UDDI Business Registry) servers potentially makes servers become the bottleneck of the system. (2) The information stored in UBR servers is static. (3) To keep the coherence and validity of the UBR servers' information is difficult. Therefore, distributed storage and independent management could be the basal ideology to improve UDDI. Agent is a conception developed in the field of the artificial intelligence these years [3]. Meanwhile, agent has been an important method to resolve the problems of distributed systems [4]. In this paper, we introduced an agent based service discovery mechanism. Its essential is that every node, which provides web services, is an agent and WSDL documents are distributed stored in nodes; A web-services searching tree composed of agents comes into being so that the service requestor can query information along it.

2 The Kernel Algorithms of ABSDM

In ABSDM, the web service management is achieved during the management of service agent (for short: SAgent). A new server connecting the network, a corresponding SAgent is created, where there are WSDL documents describing the web services in the new server. The SAgent structure is described in Backus-Naur Form as follows:

[1] The Project was supported by Zhejiang Provincial Natural Seience Foundation of China. No. 602045 and M603169

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 441–444, 2004.

SAgent ::= <function module><information library>
function module ::= <Information library management><send/receive service request ><receive result>
Information library ::= "local SAgent information" <Service description>
"father-SAgent information" <Service description>
{["son-SAgent information "<Service description>]}
Service description ::= <Web service interface><Web service implement>
Service interface ::= <type><message><operation><port Type><binding><port>
Service implement ::= <service>

The service interface and implementation are described with a WSDL document, and the format is explained in W3C note [5].

According to the order of SAgent joining to the network, we could create a web-service-searching tree. The Searching-Tree Creating Algorithm (STCA) could be divided into several steps as follows: (1) SAgent creating. When new service was published to Internet, a new SAgent should be created according to the format mentioned. (2) SAgent registering. At beginning, the new SAgent starts searching a whole tree, finding the nearest SAgent as its father-Sagent, and registering in it. If there were no existing searching tree, the new SAgent would become the root of a new tree. During registering, services and requests that the father could provide would be add to the information library of the son. If there are several SAgents that are equidistant to the new SAgent, it will choose one as it father-SAgent randomly. (3) Leaving. At first, the SAgent tending to leave will ask its father to remove its information from the father's information library. After that, the son will choose one of his sons randomly as its replacer and send the replacer's information to the father. Then, the son will ask all but the replacer of its sons to register to the replacer. (Certainly, if the leaving SAgent has no father, it just needs to do the last step.)

Based on the tree structure, A Service Discovery Algorithm (SDA) was brought forward for service discovery. SDA could be divided into several orderly steps: ①The SAgent received a search request will search its information library at first. ② If there is no target document in local information library, the searching request will be referred to the whole son tree of current SAgent.③ If there were no target document in current SAgent and its sons, it would refer the request to its father. The father will repeat the same operation in ①. If current SAgent is the root of the searching tree, searching attempt will fail.

Compared with UDDI, the characteristics of ABSDM are presented as follows: i) Managing resource dynamically. In ABSDM, service managing and publishing was dynamically implemented during the process of SAgent creating and registering. (ii) Distributing deposited service information. There is no so much information in SAgent as in UBR. Further more, service providers in UDDI should login on a UBR server to maintain service information, while every SAgent could maintain information locally. (iii) Service searching synchronously. As described in SDA, searching request could be distributed by a SAgent to its sons and performed there synchronously. In this way, searching request could be synchronously performed in many SAgents.

3 An Example

We will illustrate how ABSDM was implemented with the following example from a project, the Application Platform of Virtual Research Center (VRC), main function of which is sharing files among VRCs. The structure of its network is showed in Fig.1 including root servers and 18 VRCs every of which is of 5 sub-VRCs. Because the files saved in nodes are just like WSDL documents, the file-sharing system could be a typical example of ABSDM.

Given that there are 10000 documents in every node, and every document is different from the others. The time used to search one in 10000 documents is denoted as T_{DB}, the time used to send searching request and result from node to node is denoted as T_{net}, T_{net} = (sending request (or result) data)/(network bandwidth). With the universality, we suppose that searching request were sent by 1A, the first sub-VRC of the first VRC, and the document that 1A requested were kept in sub-VRC 18E. The whole searching process is shown in table1. The time every step spending is also listed.

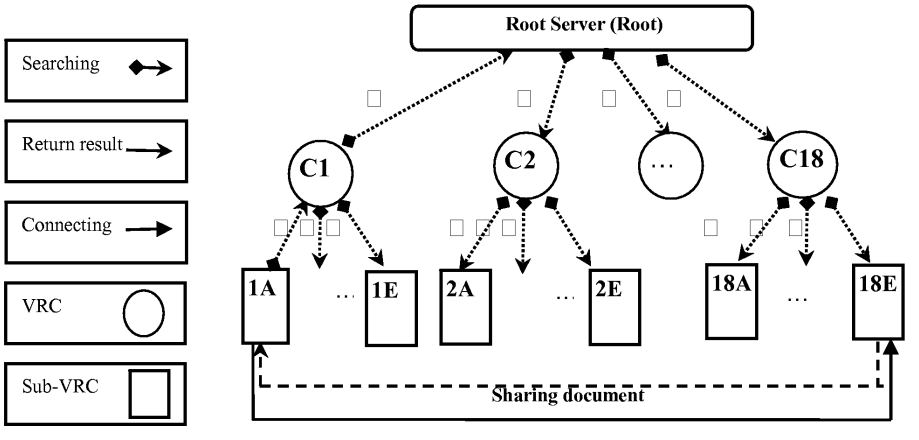

Fig. 1. An example

In this example, we could suppose that there is equal probability for every document to be requested. Because that the total number of document (denoted as totalnum) = (node number)*(document number in every node) = (1+18+18*5)*10000=1090000, then: (the occurred probability of every step) = (searched documents number in this step)/totalnum. The result is listed in Table 1. Based on the result, we could compute the average time cost: $T_{ABSDM} = \sum_{cond=1}^{6} T_{all} = 9.27 T_{net} + 5.63 T_{DB}$.

If the file-searching process is in the way of UDDI, the time cost is $T_{UDDI} = 2 T_{net} + T_{DB}'$, the mark T_{DB}' denotes the time used to search one in 1090000 documents. From experiment, we get the value of T_{net}, T_{DB}, T_{DB}' as follows: T_{net} = 20ms, T_{DB} = 28ms, T_{DB}'

= 532ms. So in our deduction in theory and experiment, $T_{ABSDM}/T_{UDDI} \approx 0.33$, ABSDM could be more efficient than UDDI.

Table 1. The process of document searching

	Document location	Probability	Document number	Time wasted			Next Nodes
				Used	Current	Totally time to the step	
Step 1	1A	0.92%	10000	0	T_{DB}	T_{DB}	VRC1
Step 2	VRC 1	0.92%	10000	T_{DB}	$2T_{net}+T_{DB}$	$2T_{net}+2T_{DB}$	Son of VRC1
Step 3	Son of VRC1	3.67%	40000	$2T_{net}+2T_{DB}$	$2T_{net}+T_{DB}$	$4T_{net}+3T_{DB}$	Root
Step 4	Root	0.92%	10000	$4T_{net}+3T_{DB}$	$2T_{net}+T_{DB}$	$6T_{net}+4T_{DB}$	Other VRCs
Step 5	Other VRCs	15.60%	17*10000	$6T_{net}+4T_{DB}$	$2T_{net}+T_{DB}$	$8T_{net}+5T_{DB}$	Son of other VRCs
Step 6	Son of other VRCs	77.98%	17*5 *10000	$8T_{net}+5T_{DB}$	$2T_{net}+T_{DB}$	$10T_{net}+6T_{DB}$	No nodes left, process ended.

Explanation: *At one time, the searching action was performed concurrently in many searching tree nodes so all of these actions just needed one T_{DB}.*

4 Conclusion

From above discussion, we can see that ABSDM could help to improve the efficiency of web service discovery while data was distributed to nodes evenly and nodes were connected by high-speed network. Further more, ABSDM could be used for reference in some fields such as distributed database, parallel computing, etc. On the other hand, when data is distributed among nodes very unevenly or nodes are connected by low-speed network, the performance of ABSDM may be poorer than UDDI. In the future, we plan to improve our idea in the following aspects: 1. To optimize the algorithm so that searching request could be dispatched into network more quickly. 2. To avoid network jam caused by searching request being copied and spread widely.

References

1. F. Curbera et al., Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, vol. 6, no. 2, Mar./Apr. 2002, pp. 86–93.
2. Wolfgang Hoschek. The Web Service Discovery Architecture. In Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC2002), Baltimore, USA, November 2002. IEEE Computer Society Press.
3. Wooldridge M., and Jennings N. R. Intelligent agents: theory and practice. Knowledge Engineering Review, 1995, 10(2):115-152
4. F. Zambonelli, N. R., Jennings,etc., Agent -Oriented Software Engineering for Internet Applications, 326-346. Springer Verlag. 2001.
5. Ariba Inc., IBM Corp., and Microsoft Corp.,Web Services Description Language (WSDL) 1.1, `http://www.w3.org/TR/wsdl`, W3C Note, 2001