DYNAMIC DISCOVERY OF WEB SERVICES USING MOBILE AGENTS

Gaurav Tiwari, Rahul Agrawal, Shakti Mishra and Dharmender Singh Kushwaha
CSED, MNNIT, Allahabad, India

Keywords: Web services, Mobile agent, Web service discovery.

Abstract: Although UDDI has been extensively been promoted for discovery of web services, it fails to provide many features like service provider validation, semantics lookup, Quality of Service Metadata, trust establishment etc. We propose a mobile agent based approach to the discovery of web services. The mobile agent framework is implemented using Java Agent Development Framework. The experimental results show that the mobile agents reduce the need for bandwidth as only data transfer required is that of the mobile agent itself. Since the mobile agents have all the data within themselves they need not communicate with other system. This also reduces the load on the system and makes the system fault tolerant.

1 INTRODUCTION

Web services are typically application programming interfaces (API) or web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services Web services provide the potential of implementing the real world business functionality, because of its flexible and adaptable framework. This flexibility has allowed Web services to become part of many existing application environment and has been one of the reason behind their popularity (Erl, 2008). Web services required an Internet-friendly and Extensible Markup Language (XML) compliant communication format that could establish a Simple Object Access Protocol (SOAP) standardized messaging framework.

Web services provide access to software systems over the Internet using standard protocols. Web Services Description Language (WSDL) is a specification defining how to describe web services in a common XML grammar (Cerami, 2002). Microsoft, IBM and Ariba proposed UDDI (Universal Description, Discovery, and Integration, 2000) to describe a standard for an online registry, publishing and dynamic discovery of web-services offered. Microsoft and IBM proposed WSDL as an XML language to describe interfaces to web-services registered with a UDDI database (Lu et al., 2002).

Publishing a Web service involves creating the software artifact and making it accessible to potential consumers. Optionally a provider can explicitly register a service with a Web services registry such as UDDI or publish additional documents intended to facilitate discovery such as Web Services Inspection Language (WSIL) documents. The service users or consumers need to search Web services manually or automatically.

Discovery of web services can be carried out in various ways and one of the way is to accomplish this by using dynamic acting entities as mobile agents. A mobile agent is a composition of computer software and data which is able to migrate from one computer to another autonomously and continue its execution on the destination computer (Miraikar, 2006).

When a mobile agent decides to move, it saves its own state, transports this saved state to the new host, and resumes execution from the saved state. Mobile agents are active entities that are free to migrate between computers at any time during their execution (Ketel, 2009). It makes these entities a powerful tool for implementing distributed applications in a computer network. Web-services are registered with some public registries, intelligent mobile agents might migrate from one service registry to another to find desirable web-services as specified by a user. Letizia (Lieberman) is a user interface agent that assists a user browsing the World Wide Web based on the user's profile.

Tiwari G., Agrawal R., Mishra S. and Singh Kushwaha D..
DYNAMIC DISCOVERY OF WEB SERVICES USING MOBILE AGENTS.
DOI: 10.5220/0003143604970500
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2010), pages 497-500
ISBN: 978-989-8425-29-4

We have designed and implemented a framework based on mobile agents for the discovery of web services. A bunch of mobile agents migrate from one machine to other collecting information about web services and reporting them.

2 PROPOSED ARCHITECTURAL FRAMEWORK

We propose a mobile agent based approach to the discovery of web services. Our approach has three major sections.

- The first deals with the discovery of web services and reporting them.
- The second deals with publishing the web services.
- Finally, the third deals with handling incoming requests for finding the best possible web service from all the available services.

2.1 Components of Architecture

Service Requester. It is the node that issues request for a certain type of service according to its requirements and it would include any or all of a list of parameters like service name, type and number of inputs, return type etc.

Request Optimizer. It tries to deliver the most similar web service matching the given requirement.

Service Broker. It is the repository of all the web services. There can be more than one service brokers in the system.

Web Service Provider. Web service providers (WSP) actually publish the web services.

Web Service Repository. The web service repository is kept in the form of a relational database. The following are the fields of the database are Hostname, Service name, Function, Input, Output, Install date, Discovery date and Report date.

Controller Agent. The controller agent is responsible for the creation and management of other mobile agents. It is also responsible for determining the course of migration of the agents.

2.2 Functions of Mobile Agents

The mobile agent performs three functions:

1. First, it sends query to all machines for the list of web services installed and compares this list with the list that it already has discovered or the changes that have been done to the old ones.
2. Second, when the agent reaches one of the designated nodes, it reports the collected data (list).
3. The agent would also collect performance data about the nodes. This includes performance metrics like network load, physical memory, processor usage so that the service requests could be matched with the most efficient machine on the network.

The migration of the agent is managed by a controller agent. The controller agent would act as the scheduler for deciding the migration of agents.

2.2.1 Discovery & Reporting of Web Services

Identifying Web Services. The mobile agent searches the local system for the installed web services & scans the interface where the WSDL of the web services are published.

Collecting Performance Data. The agent also collects performance Metrics. These performance metrics would help the request optimizer to find the best match for an incoming request.

Report the Web Services. When the agent arrives at the designated node for the reporting of web services, it updates the local database.

2.2.2 Publishing a New Web Service & Updating Existing Web Service

The node deciding to publish a new web service would first create its description using the standard WSDL and develop the corresponding implementation for the service, thereby publishing the description.

2.2.3 Finding the Closest Match to the Discovery Request

The service broker has the list of all the web services with it. It matches it with the best possible service that it has in its repository. If there are more than one web services matching the request, the service broker can reply back with a list of all of these services to the requester. However, the broker can break the tie on the basis of the performance criteria collected by the agent.

2.3 Migration

The mobile agent uses the services of the Controller Agent (CA) in order to decide the next machine to migrate to. It calls the Controller Agents scheduling algorithm with its current location. The controller agents decide the next location of the calling agent (fig. 1).

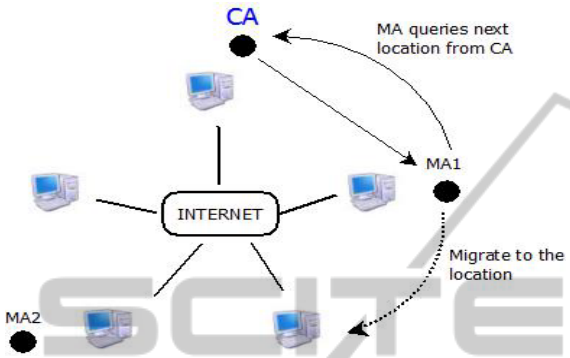

Figure 1: Migration of Mobile Agents.

Round Robin Migration. Under this scheme the agent sequentially migrates through each and every node in the system one by one (Fig.2).

```
ROUNDROBIN-next-destination
(currentnodeID,locationlist)
{
    next-destinationID = curentnodeID+1;
    next-destinationID%=sizeof(locationlist)
    return next-destinationID
    }
```

Figure 2: Round Robin Algorithm for Migration of Mobile Agents.

Most Active Next Migration. This migration algorithm (Fig.3) visits the most active nodes more than the less active node. The controller agent maintains a count of the web services created and/ or modified by all the machines. To select the next destination, the controller agent uses a priority queue. To avoid starvation, relatively higher weight should be given to time since last visit**.**

3 TOOLS AND FRAMEWORKS USED

We have used Apache Axis, which is an open source, XML based Web service framework. It consists of a Java and a C++ implementation of the SOAP server, and various utilities and APIs for generating and deploying Web service applications. The mobile agent framework is implemented using JADE (Java Agent Development Framework) fully implemented in Java language.

```
Global: priorityArray[ ].
Initialize priority Array with all having zero priority.
MOST ACTIVE NEXT(currentid,locationlist)
{
  //update priorityArray
  For each location in location list
      Find the percentage of recent change
      performed by it
      Calculate the time since any agent had
       last visited the node
      update the priority in the priorityArray
destinationID=index with maximum priority
  return destination ID
  }
```

Figure 3: Most active Next Migration Algorithm.

3.1 Creation of JAVA Web Service

In our proposed system, a Web Service is a JAVA Class. The public functions of the class would be available when the service is published.

3.2 Deployment of Web Service

Once the WSDL and Web Service are ready, we create a deployment descriptor. The deployment descriptor describes the method to deploy the web service. In the deployment descriptor, we allow or disallow routines in the web service.

4 EVALUATIONS

For each set of values, we have calculated the average delay. To simulate different sequences of events such as installation, discovery time and reporting time, we take different number of nodes to manage varying number of agents.

4.1 Computing Effect of Number of Agents on the Average Delay between Installation and Discovery Time of Web Services

We observe from fig. 4 that as the number of machines increase, the candidates for inspection by the mobile agent increases. We can see that the increase in number of machines causes the average

delay between installation and discovery time of the web services to increase. As the number of agents increase, the workload for one agent decreases. As the agents work asynchronously and parallel, the web services are discovered in shorter span of time.

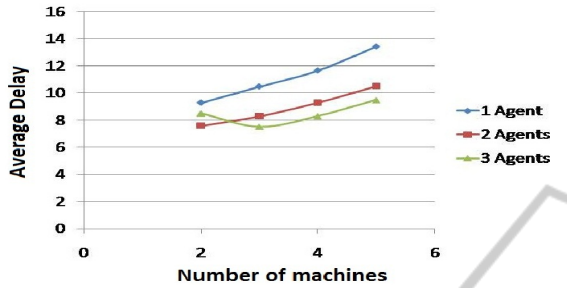

Figure 4: Effect on number of agents on the average delay between installation and discovery of web services.

4.2 Average Delay between Installation and Reporting Time of Web Services

We observe from fig. 5 that as the number of machines where the data has to be reported increase, the agent would have to stop at more location to submit the data. This causes an increase in the average delay between installation and report time.

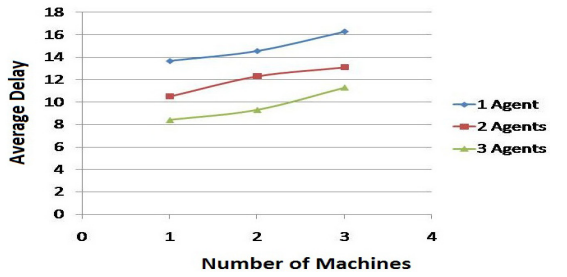

Figure 5: Effect on number of agents on the average delay between installation and reporting of web services.

4.3 Average Delay in Installation and Discovery

We observed from fig. 6 that as the number of web services increases, the agent requires more time to check for the new services or the changes in the old ones. Thus, the average delay increases as there are more services to be checked and reported.

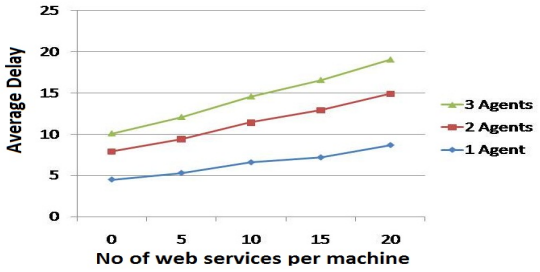

Figure 6: Effect on total number of web services on the average delay in installation and discovery.

5 CONCLUSIONS

This paper presents the mobile agent based approach to the discovery of web services. Although UDDI has been extensively been promoted for discovery of web services, it fails to provide many features like service provider validation, semantics lookup, Quality of Service Metadata, trust establishment etc. We have shown that the mobile agents reduce the need for bandwidth as only data transfer required is that of the mobile agent itself. We have also shown that the mobile agent based system is fault tolerant. As there are more than one mobile agent in the system, even if some fail the others could complete the task and would not allow the system to go down. Thus the use of a mobile agent based approach is a good alternative to the conventional approach using centralized servers to report web services.

REFERENCES

Bellifemine, F, et Al. *JADE A White Paper.*

Pautasso, C., *Limitations of SOAP, WSDL, UDDI.* Computer Science Department, Swiss Federal Institute of Technology.

Gengler, B., 2001. *UDDI Has Problems . Computer Fraud & Security.* Volume 2001, Issue 8

Chapell, D., article_*who_cares_uddi.html.* http:// www.davidchappell.com/articles/

Lu, S., Dong M. and Fotouhi, F., 2002.*The Semantic Web opportunities and challenges for next-generation, Web applications ", Information Research, Vol. 7 (4)*

Lieberman, H. *Letizi An Agent That Assists Web Browsing.*

Erl, T., 2008. *Service-Oriented Architecture.* Pearson Education

Cerami, E. , 2002. *Web Services Essentials,* O'Reilly

Miraikar, Z., 2006. *Resource and Service Discovery for Mobile Agents Platform.*

Ketel, M. 2009. *A Mobile agent Based Framework for Web Services", IEEE Internet Computing V. 10(3),* pp. 58-65.