TOWARDS AN AGENT ENABLED GRID ENVIRONMENT

Nguyen Xuan Vinh, Le Thi Phuong,

Nguyen Thanh Thuy, Le Dang Hung, Dao Tran Minh

High Performance Computing Center
Hanoi University of Technology
N^o 1 Dai Co Viet Street – Hanoi – Vietnam
VtheSniper@yahoo.com, hn_hut@yahoo.co.uk
{thuynt,hungld,minhdt}@it-hut.edu.vn

Abstract. *In Grid environments, the agent technology has recently emerged as a promising tool to solve complex and difficult problems such as resource brokering, resource scheduling and grid application development. The question is how to integrate the agent technologies within grid frameworks. Different solutions have been proposed but this question is still open. This paper presents a new solution called "hybrid agent – grid service" (HAGS); the idea is to develop grid services by extending characteristics of agents. This will facilitate the communication and interaction between grid services and agents in a grid environment. The strong point of this solution is the reuse, which means that people can reuse existing agent and grid framework (Jade and GT) rather than to develop a new one supporting both grid and agent technologies.*

Keywords: Multi Agent systems; Grid middleware; Hybrid agent; Grid service.

1 INTRODUCTION

In Grid environments, Agent technologies have recently emerged as a promising tool to solve complex and difficult problems, e.g. resource discovery and monitoring, workflow management [3,4]. Agent technologies provide autonomous and flexible problem solving capabilities in uncertain and dynamic environments such as a Grid environment. One of the complex issues in realizing a Grid environment is to manage heterogeneous and distributed resources, e.g. computing and storage elements, and software components. An autonomous solution that supports negotiation between resource users and resource providers, and between domains using different resource allocation and security policies, is highly demanded [1,2]. Agent technologies encapsulate control intelligences and provide adaptable solution to solve problems collaboratively.

However, integrating Agents with Grid environments is interdisciplinary and requires knowledge from different domains such as distributed systems and artificial intelligence. One of the central problems is how to enable Agents to interact seamlessly with the grid services in a Grid environment . The Grid Computing Research Group at the High Performance Computing Center (HPCC) of Hanoi University of Technology (HUT) has been working on this domain for several years. This paper presents some of our recent results about an agent system called Hybrid Agent – Grid Service System which has been prototyped in BKGrid 2005, a grid system developed at HPCC - HUT.

The rest of this paper is organized as follows. In section 2 we introduce the background of Agent and Grid middlewares and describe our research mission. After that we review the related work and discuss the basic idea of the hybrid agent – grid service architecture. The design and implementation are discussed in Section 4. Some applications of the hybrid agent – grid service system in BKGrid 2005 is presented in Section 5.

2. STATE OF THE ART

2.1. Globus toolkit

Globus toolkit is the leading open source toolkit used today to build Grid enviornments. It is built and maintained by the Globus Alliance, an organization composed by many institutions and companies [13]. The open source Globus Toolkit is a fundamental enabling technology for the "Grid" letting people share computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security, file management, data management, communication, fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop applications. Every organization has unique modes of operation, and collaboration between multiple organizations is hindered by incompatibility of resources such as data archives, computers, and networks.

The Globus Toolkit was conceived to remove obstacles that prevent seamless collaboration. Its core services, interfaces and protocols allow users to access remote resources as if they were located within their own machine room while simultaneously preserving local control over who can use resources and when. The Globus Toolkit has grown through an open-source strategy similar to the Linux operating system's, and distinct from proprietary attempts at resource-sharing software. This encourages broader, more rapid adoption and leads to greater technical innovation, as the open-source community provides continual enhancements to the product.

The toolkit we are currently using is Globus Toolkit version 3.2 (GT3.2). GT3.2 is an implementation of the OGSI (Open Grid Service Infrastructure) and OGSA (Open Grid Service Architecture) standards. It follows a service-oriented architecture. The core idea in GT3.2 is grid services, an extension of the current widely used Web service technology.

2.2. JADE Agent frameworks

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language [12]. It simplifies the implementation of multi-agent systems through a middle-ware that complies with FIPA specifications [14] and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in Java language and the minimal system requirement is the version 1.4 of JAVA (the run time environment or the JDK).

3. INTEGRATING JADE AGENTS INTO GLOBUS BASED GRID ENVIRONMENTS

We start by stating the main difficulties in integrating Jade Agent into Globus Grid environments. As for the JADE framework, to build an agent, we must extend the Agent class and then setup some behaviors for our agent in term of methods. Similarly with the Globus Toolkit, to build a grid service, we have several options: one is to extend directly the GridServiceBase class and then setup the methods for our service; another way is using operation providers and our Grid service will delegate tasks to these operation providers (Globus Toolkit is mainly based on Java).

One of the main technical difficulties in integrating Agent into Grid environment lies in the fact that Agents and Grid Services live in separate environments: JADE Agents need to exist in the JADE Agent platform; Globus Grid Services must live in the Globus Grid Service Container or some other kinds of hosting environments. This means that Agents and Grid Services live in different address spaces. Within the agent community, agents talk with each other using standard messages. In the grid community, grid services talk to their clients as well as to other grid services using standard messages too, of course in a different format. So how Agents and Grid Services can talk to each other. What we need is a solution that can ease the communication and interaction between objects living within these two separate environments.

3.1. Related work

There has been several approaches to integrate agent into grid. In the SoFAR project [6,7,11], the authors provide a method to align Agent technology with Web service technology. They have shown how an agent system could integrate its transport layer with Web Services [6]. According to the SoFAR's approach, the agent behaviors can be described, advertised and discovered using Web Services technologies. WSDL (Web Service Description Language) is used to describe agent behaviors, which can then be advertised in UDDI (Universal Description, Discovery, and Integration) registries as regular services. Additionally, they provide an extended UDDI registry, able to support advanced queries used for discovering in agent system.

In [10] the authors have mentioned two different ways to realize an agent – grid system: i) to extend grid middleware to support agent features and ii) to extend agent-based middleware to support grid features. In that paper they followed the second way by extending the JADE agent framework to support grid features such as mechanisms for code distribution, reconfiguration, goal delegation, load balancing optimization and QoS definition.

3.2. A hybrid agent – grid service (HAGS)

In this paper we provide another solution: a Hybrid Agent – Grid Service (HAGS), which is somewhat in the reverse direction as compared to the solutions described above. The idea is to extend grid middleware to support agent features. We do that by leveraging existing grid and agent framework namely JADE and Globus. In the SoFAR project [6,7], the authors consider an agent as a service: they use Web service facilities to expose agent features. Our solution considers a grid service as an agent and add agent's behaviors into grid service. A grid service can expose its functions through the agent's broadcasting capability. Vice versa, an agent can also expose its functions through the grid information system.

With this solution, the complicated problems of communication and interaction between the Grid and Agent community can be solved completely. Agent will be integrated seamlessly into the Grid environment: each agent will be a grid service and vice versa each grid service can act like an agent. Therefore, each time an agent want to talk with other grid services in the grid, it simply acts like a grid service, and each time a grid service need to interact with the agent community, it simply acts like an ordinary agent. In the next sections the implementation details of the solution will be outlined.

4. DESIGN AND IMPLEMENTATION

So how can we do that exactly? The core idea seems to be very simple: writing a grid service that extends the Agent class.

4.1. Preparation

As stated above, the objectives is to create grid services that extend the Agent class so it is necessary to make Globus recognize the classes related to the JADE Agent platform. For this purpose, the following JADE libraries must be copied to the Globus Toolkit library directory (I.e. $GLOBUS_LOCATION/lib on our Linux computer with the $GLOBUS_LOCATION environment variable points to the setup directory of Globus) : jade.jar, iiop.jar, crimson.jar, base64.jar, jadeTools.jar and http.jar into the Globus toolkit library directory. These are the basic libraries required to run Jade. By this way, first, when we write and compile our HAGSs, the compiler will know where agent libraries lie; and second, when we start our HAGSs, the container will know where to start the agent platform as well as invoke agent methods.

That's all the preparation tasks we need. From now on we can start experimenting with our Agent – based grid services. There are two kinds of agent-based grid services to be created. One is the Auto Setup Agent - grid service that will automatically start up the whole agent platform each time we start the grid service container, and the other is the ordinary HAGSs that will join the agent platform pre-created by the Auto Setup Agent.

4.2 The Auto Setup Agent grid service

Readers who are familiar with JADE agent programming certainly have known that before we can start a new agent, we need to start the JADE agent platform (just as we must start the Globus grid service container or some other kinds of hosting environment before we can use grid services). The simplest and most straightforward way to do that is to launch a command line. I.e. *java jade.Boot –gui* to start the JADE agent platform on that computer with a graphical interface. But this approach is rather awkward since each time we start our grid service container we must manually start the JADE agent platform before to ensure that our hybrid agent – grid services created later will have somewhere to live. So we tried to develop a so-called Auto Setup Agent grid service that automatically setup the whole JADE agent platform along with the grid service container.

This service is set as a persistent grid service which automatically loads the Jade agent platform as the grid service container is being started. Another important thing which must be ensured is that the Auto Setup Agent grid service must be started before all the other hybrid agent – grid services. This can be done by editing the server configuration file of Globus: move the XML block that describes the Auto Setup Agent grid service above all the other grid services. And finally when the grid service container is going to be shutdown, the Auto Setup Agent grid service must be able to recognize this and shut down the agent platform automatically as well.

The detailed steps involved in writing the Auto Setup Agent grid service is quite tricky since we had to dig through the source code of JADE and extract some codes for our purpose.

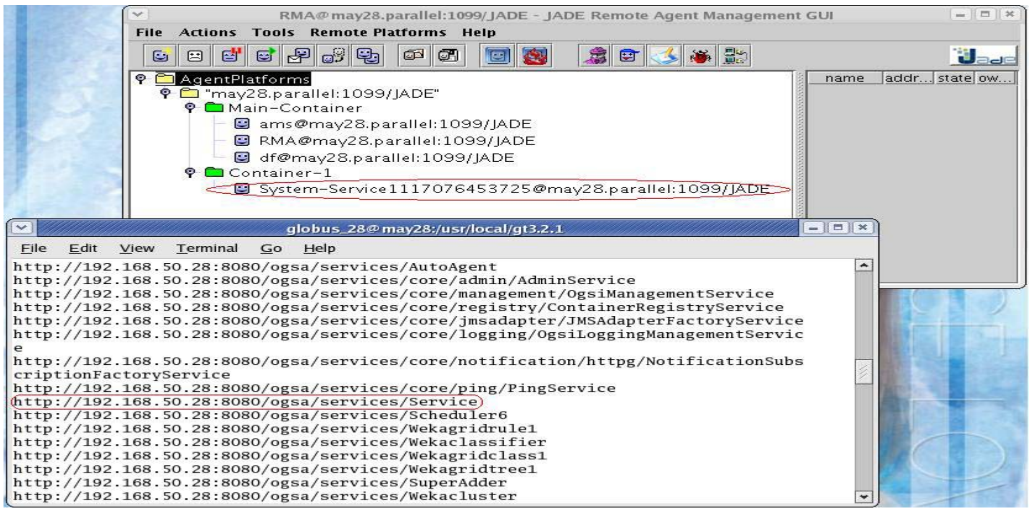
But repeating this process seems not necessary since readers can use our ready-to-use Auto Setup Agent grid service. Once the system has been setup properly, we are ready to write our hybrid agent - grid services.

4.3 The other hybrid agent – grid services

As said earlier, every agent must extend the Agent base class, so in order to convert grid services into HAGS we must implement grid services using operation providers mechanism. It's the operation provider classes that will extend the Agent base class and act like an agent. There are several important methods:

- In the initialize() method of the operation provider, we must create a non-main container and connect it to the default main container of the JADE Agent platform stated on this machine by the Auto Setup Agent grid service or by manual methods. The hybrid agent – grid service instances will live within this container.
- The setup() and takeDown() methods are inherited from the Agent base class and are automatically called by the JADE Agent platform. For these methods we just setup code just as for ordinary agents.
- One odd characteristic of hybrid agent – grid service is that when we are done with its instance, the agent still alive even when we explicitly call the destroy() method of that grid service. This is because as we stated above, the hybrid agent – grid service exist in both environment: the grid service container and the agent platform. Hence for a hybrid service to automatically destroy itself completely, we must implement the callback interface in our grid service and order the agent to delete itself when the grid service is going to be destroyed.

The other grid service methods are set up just as for normal grid services. Once we have finished our grid service, we can deploy it as usual. The next time we start the grid service container the hybrid agent – grid services will appear in both the agent platform and the service container as seen in figure 1.


RMA@may28.parallel:1099/JADE - JADE Remote Agent Management GUI
File Actions Tools Remote Platforms Help
name addr... state ow...
globus_28@may28:/usr/local/gt3.2.1
File Edit View Terminal Go Help

```
AgentPlatforms
  "may28.parallel:1099/JADE"
    Main-Container
      ams@may28.parallel:1099/JADE
      RMA@may28.parallel:1099/JADE
      df@may28.parallel:1099/JADE
    Container-1
      System-Service1117076453725@may28.parallel:1099/JADE
```

```
http://192.168.50.28:8080/ogsa/services/AutoAgent
http://192.168.50.28:8080/ogsa/services/core/admin/AdminService
http://192.168.50.28:8080/ogsa/services/core/management/OgsiManagementService
http://192.168.50.28:8080/ogsa/services/core/registry/ContainerRegistryService
http://192.168.50.28:8080/ogsa/services/core/jmsadapter/JMSAdapterFactoryService
http://192.168.50.28:8080/ogsa/services/core/logging/OgsiLoggingManagementServic
e
http://192.168.50.28:8080/ogsa/services/core/notification/httpg/NotificationSubs
criptionFactoryService
http://192.168.50.28:8080/ogsa/services/core/ping/PingService
http://192.168.50.28:8080/ogsa/services/Service
http://192.168.50.28:8080/ogsa/services/Scheduler6
http://192.168.50.28:8080/ogsa/services/Wekagridrule1
http://192.168.50.28:8080/ogsa/services/Wekaclassifier
http://192.168.50.28:8080/ogsa/services/Wekagridclass1
http://192.168.50.28:8080/ogsa/services/Wekagridtree1
http://192.168.50.28:8080/ogsa/services/SuperAdder
http://192.168.50.28:8080/ogsa/services/Wekacluster
```

Figure 1: The hybrid agent – grid service appears in both the grid service container and the agent platform

Figure 2 shows the hybrid nature of a HAGS. The hybrid agent – grid service system above has many nice features. The agent platform is integrated seamlessly into the grid environment just like a part of it. We do not have to concern about any thing about the agent platform below and develop our hybrid services very easily. The hybrid agent – grid service has a no limited capability of communicating and interacting with entities in both environments. The whole system is very compact and the implementation is quite simple yet provide great facilities for developers who want to experiment with agent in grid but do not have deep knowledge about both technologies. In the next sections we will describe some applications of Agent technology in our grid at Hanoi University of Technology using the techniques above.

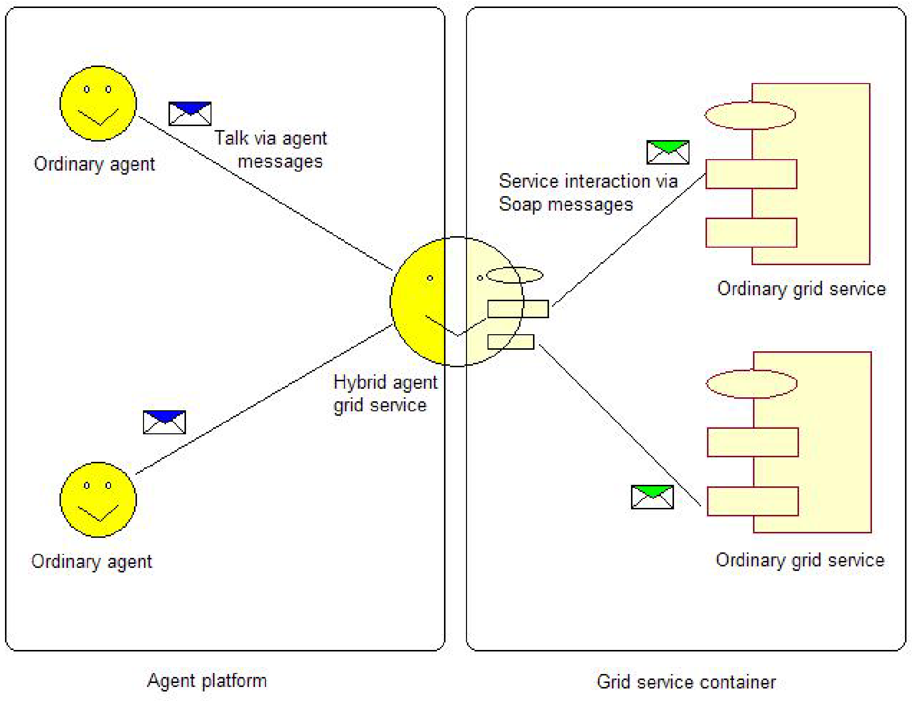


Figure 2: Hybrid agent can smoothly interact with entities in both environments

5. APPLICATIONS OF AGENT TECHNOLOGY IN BKGRID 2005

5.1. BKGrid 2005

BKGrid 2005 is setup at Hanoi University of Technology for research purposes. Our grid consists of dozens of PCs, workstations, servers and several Beowulf PBS based clusters and is build based upon the GT3.2 toolkit. We follow the economics-based model for grid architecture [3][4]. Our main application of Agent technology in HUT grid lies in the area of Grid resource management. We proved the agent's ability to ease the process of resource discovering; negotiating between different domains, between resource users and resources providers; and QoS control.

5.2. Application

For an experimental grid application, we picked up Weka, a well-known software kit for data mining. In [5], the authors provided an ad-hoc solution to enable Weka for the grid environment: they do not use any standard Grid toolkit but rely mainly on pure Java and its network libraries. In our implementation, to take the most advantages of the existed code, we just extracted and rewrote main Weka functions in grid services, that means, created a grid service interface for existed Weka functions. This implementation doesn't make much sense if the user has only one job (i.e. one data file) to run: the job still runs on only one machine. Therefore we created a parameter sweep-like application which enables users to submit a set of files. The Weka toolkit also provides a tool called WekaParallel for data mining on a cluster of computers. We are creating grid interface for this tool just like for the above Weka functions. By this way our so-called Data mining Grid will have a variety of resource from single PCs to clusters. Each Weka service is also implemented in term of an Agent. Each time an instance of this service is created, this agent will come to exist and register itself with the local agent platform's DF agent (Directory Facilitator agent). By that way the resource manager agent on that machine can count and control the maximum concurrent instances allowed, therefore control the QoS for users.

5.3. The multi agent system as a grid information service

In our system the multi agent system can be used as a supplement for the Grid Information Service (GIS) provided by GT3.2. Each hybrid agent – grid service has a description of the functions it can provide as well as other fields of information. Those agents will register themselves with the DF agent on the local machine or on another machine on the grid serving as a Grid Information server. When one needs to find a specific service he or she can use the great capability of searching and matching provided by the agent system.

5.4. Agent as a resource manager

In each of resources of the system such as PC, server and cluster, we deploy a System Agent that serves as a resource manager. The system agent can provide information about the resource which include static and dynamic information such as system architecture, operation systems, CPU load, free ram... which can be retrieved via functions in the GT3.2 core library.

As a resource manager, the System Agent decides which user can use a specific service at a given time. In our system, every user who wants to use a service must first acquire the grant for using that service from the System Agent. To assure QoS, the System Agent will control the number of instances of other agent – grid services. This number depends on the capability of that resource, or more exactly, on the policy of that resource owner. The user asks for an instance of a specific service from the System Agent. If this is acceptable, the System Agent will create a new instance of that service and inform the user details about the newly created service. Each instance of the hybrid agent – grid service is also an agent and it will register itself with the local Agent platform's DF agent so that the System Agent can count and control the number of instances of each service type.

The System Agent also provides basic advance reservation functions. It maintains a time slot table. The user uses his own grid ID to request an advance reservation for a specific period of time in the future. The System Agent will check whether during that period of time the number of users has exceeded the maximum number of users that resource can serve concurrently. If not, it writes the user's grid ID in that time slot table and the user has successfully reserved that resource for future use. When a user has a request for that resource, the System Agent will consider the number of reserved users as well.

5.5. Agent as a resource broker

Users use services through their resource brokers. Each user has his own resource brokers for submitting the jobs. According to the economic model describes in [3], [4] and many other papers, the brokers act quite independently and compete for using resources. The resource broker is implemented in term of an agent – grid service. It interacts with the agent system to discover and find a suitable resource according to user's requirements on time and budget limitation. The broker agent will then negotiate with the System agent on a specific resource for using that resource. Once the choosing and negotiating process has been done successfully, the broker, as a grid service, will call the grid service on that resource on behalf of the user.

Figure 3 shows a probable interaction scenario between entities in an agent enabled grid environment. During its lifetime, the Resource Manager Agent (the System Agent in our implementation) registers and periodically renews its information with one (or some) DF agent(s) in the grid served as the market directory agent. Its information includes resource information and description of the services on that resource. The broker agent, on behalf of its user, will query one (or some) DF agent(s) to locate a desired service. After that, the broker negotiates directly with the Resource Manager Agent for resource grant. The Resource Manager Agent will then count the instances of that service and check its allocation table to see whether the total number of instances has exceeded the maximum number allowed. If not then the broker may be granted to use that service. It will create a new instance of the HAGS and call its methods. This instance will register itself with the local DF agent and deregister itself when the broker has done.

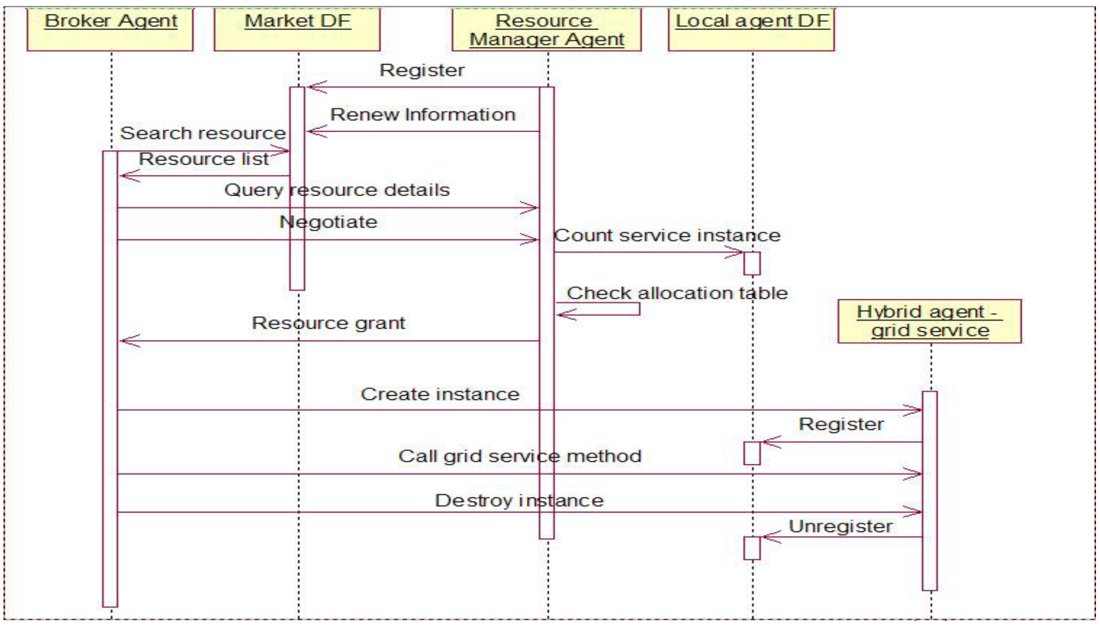


Figure 3: An interaction scenario between hybrid agent – grid services and other grid and agent entities

6. CONCLUSIONS

This paper shows a practical solution for the issue of integrating agents into a grid environment. We presented the idea of a hybrid agent – grid service (HAGS) along with the technical details needed to realize a complete HAGS system. The technique is quite simple yet it exploits the great facilities provided by JADE and Globus to create a flexible solution. This technique has been implemented in the BKGrid 2005 grid system at HPCC – HUT. We have been able to solve most of the complex problems arising in the interaction between agents and grid services.

However, this approach still has some drawbacks: we did not really control the agent communication and this may pose some security issues. In [6] the authors provide a way to corporate the agent transport system with web service transport system and this can eliminate many security issues. The other drawbacks include the overhead caused by the whole agent system to the grid environment.

Our future work includes experimenting the hybrid agent – grid service system in a larger grid testbed as our grid at HUT grows larger as well as extending agent applications. We also intend to perform an estimation of the overhead caused by the agent system to the grid environment.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Zhiming Zhao at University Van Amsterdam for his revision of this paper as well as for his invaluable advice and comments.

8. REFERENCES

[1] Ian Foster, Nicholas R. Jennings, Carl Kesselman: *Brain Meets Brawn: Why Grid and Agents Need Each Other,* Proceeding of the 3rd International Conference on Autonomous Agents and Multi – Agent Systems, New York, USA, 2004.

[2] Isaac Chao, Ramon Sanguesa, Oscar Oardaiz: Grid Resource Management using Software Agents: *Grid Resource Management using Software Agents,* Ercim News No 59 Oct 2004.

[3] Rajkumar Buyya: *Economic-based Distributed Resource Management and Scheduling for Grid Computing,* Doctor thesis, 12 Apr 2002 Monash university Melbourne Australia.

[4] Srikumar Venugopal, Rajkumar Buyya, Lyle Winton: *A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids,* Proceedings of the 2nd workshop on Middleware for grid computing Toronto, Ontario, Canada, Pages: 75 – 80 2004 ISBN:1-58113-950-0

[5] Rinat Khoussainov, Xin Zuo and Nicholas Kushmerick: *Grid-enabled Weka: A Toolkit for Machine Learning on the Grid,* Ercim News No 59 Oct 2004.

[6] L. Moreau. *Agents for the Grid: A Comparison for Web Services (Part 1: the transport layer).* In Proc. IEEE International Symposium on Cluster Computing and the Grid, Berlin, Germany, 2002.

[7] A. Avila-Rosas, L. Moreau, V. Dialani, S. Miles, X. Liu. *Agents for the Grid: a Comparison with Web Services (part II: Service Discovery).* In Proceedings of Workshop on Challenges in Open Agent Systems (AAMAS02), pp. 52-56, 2002, Bologna, Italy.

[8] Junwei Cao, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd: *Grid Load Balancing Using Intelligent Agents,* Future Generation Computer Systems Volume 21 , Issue 1 (January 2005) Pages: 135 – 149, 2005 ISSN:0167-739X

[9] Luc Moreau et al: *On the Use of Agents in a BioInformatics Grid* Proc. of the 3rd IEEE/ACM CCGRID '2003.

[10] Agostino Poggi, Michele Tomaiuolo and Paola Turci Dipartimento di Ingegneria dell Informazione Università degli Studi di Parma: *Extending JADE for Agent Grid Applications* Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04) - Volume 00

[11] Luc Moreau, Nick Gibbins, David DeRoure, Samhaa El-Beltagy, Wendy Hall, Gareth Hughes, Dan Joyce, Sanghee Kim, Danius Michaelides, Dave Millard, Sigi Reich, Robert Tansley, and Mark Weal: *SoFAR with DIM Agents: An Agent Framework for Distributed Information Management*. In The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, pages 369 - 388, Manchester, UK, April 2000.

[12] Java Agent Development Framework available at http://jade.tilab.com/

[13] Globus Toolkit Documentation available at http://www.globus.org

[14] Foundation for Intelligent Physical Agents http://www.fipa.org