Grid Resource Discovery using Tree Data Structure for Multi-Trait Requests

Leyli Mohammad Khanli[1], **Saeed Kargar**[2], **Ali Kazemi Niari**[2]
[1] CS Department, University of Tabriz, Tabriz, Iran
[2]Department of Computer Engineering, Islamic Azad University, Tabriz Branch, Tabriz, Iran

Abstract *- Grid is an extensive environment in which different resources are dispersed geographically. A user may need a resource or a combination of resources in order to solve a problem. The task to find such a resource is borne by resource discovery algorithms. Therefore, the resource discovery algorithms are of high importance in grids. The methods proposed to resource discovery so far have not suggested a method to discover several resources simultaneously in the form of a request.*

In this paper, we have proposed a method that is able to discover simultaneously the desired number of the resources for the user. In our proposed algorithm, the cost of the resource discovery is very low. By means of this method, a user will be able to request several resources simultaneously in one format.

The results of simulations indicate that fewer numbers of nodes meet in the resource discovery stages in this method than that in the other suggested methods. Compared to other methods, this method also creates less traffic in the network.

Keywords: Grid, Resource Discovery, Multi-Trait Requests

1 Introduction

Grid is a new technology that enables the users to share different resources from long distance by using network and communication infrastructures. These resources can be heterogeneous and far from one another geographically [1]. Different methods have been suggested for resource discovery. Centralized methods [2-4] are among the methods that have been used. These methods have a central server that manages all nodes. In such environments as grids where there is a large number of users, there has been mounted a bottleneck in the server region, which reduces the system efficiency. The other methods are decentralized. There is not a centralized server in these methods which can manage all nodes. Flooding-based and Random-based are instances of this method. Although these methods have removed many faults of the previous methods, the system efficiency reduces with the increase in the number of nodes and with the variation in the resources.

Recently, there have been introduced distributed methods that use tree structure for resource discovery. These methods are more optimal in terms of the number of the produced traffic, etc. However, in none of these methods occurs the discovery of several resources simultaneously in one format. “A resource discovery tree using bitmap for grids” [5] and “FRDT: Footprint Resource Discovery Tree for grids” [6] and the methods proposed in [7-8] are instances of this method.

This paper proposes a method for resource discovery that uses a weighted tree structure as the method [6] does with this difference that the former makes it possible for the user to search for several resources simultaneously. The simulations show that the algorithms suggested in this paper find one or more resource for users without recourse to unnecessary and extra nodes, creating less traffic.

Below are discussed some of the works done with regard to the resource discovery so far. Section 3 is concerned with the explanation of the method suggested in this paper. Section 4 is associated with the results of the simulations. Finally, section 5 concerns Conclusion and further studies.

2 Related work

Various methods have been proposed as regards the resource discovery in the grid. Below are presented some of these methods.

Matchmaking is one of these methods [9] in which matchmaking service find a correspondence between requests and entities. Most methods use this algorithm [10-13].

Another group of methods uses a Semantic Communities among the nodes in the grid [14-17].

Juan Li. [18] has proposed a resource discovery method based on the Semantic Communities. In this method, a Semantic structure is used to group the similar nodes; therefore, the request for the resource discovery is sent to the related nodes only.

There is another method suggested recently for the resource discovery which makes use of tree structure [5]. A series of bitmaps have been used in the nodes. Upon the resource discovery, the user's requests are transformed to these formats and delivered to one of the nodes existing in the environment. These nodes utilizes AND operation to discover the resources required by the users.

In the previous work by the authors [6], a weighted tree structure had been used for the resource discovery. In this method are used a series of bitmaps that maintain the path to

the target in addition to keep the information of the resources existing in the environment.

In contrary to all previous methods, the method proposed in this work is able to discover a combination of the resources for the users at the same time. Another advantage of this method is that it is able to perform resource discovery without recourse to unnecessary nodes.

3 Our proposed method

As mentioned earlier, this method is based on a weighted tree structure. The information of the resources in the nodes will be stored in the form of a tree data structure called "*Resource-Tree*" (RT). Through RT, the information of the combined resources will be stored in the nodes, and the user's requests will be guided to the appropriate paths in the environment. To get more familiar with this method, the general structure of RT and the format that is stored in each field RT will be discussed in the later subsection, and then the resource discovery will be discussed in next subsections.

3.1 Resource-Tree (RT)

As pointed out before, the method proposed in this paper uses a tree data structure called RT. The size of RT depends on the type of the resources in the environment. RT includes fields in which the information related to the local node resource and/or the information of the children of this node will be stored. Fig. 1 shows an instance of RT. This RT is devised for an environment which shares 3 kinds of Operating Systems (OS) and 2 kinds of RAM. It is noted that the general structure of RT is known for all nodes in the environment. Not all nodes in the environment will use all fields in RT, but they will use some of these fields depending on the resources at hand. A sample of field RT is shown in Fig. 2. This field consists of two columns called "*Resource*" and "*Children*". The meaning of the numbers stored in these columns is explained through an example.

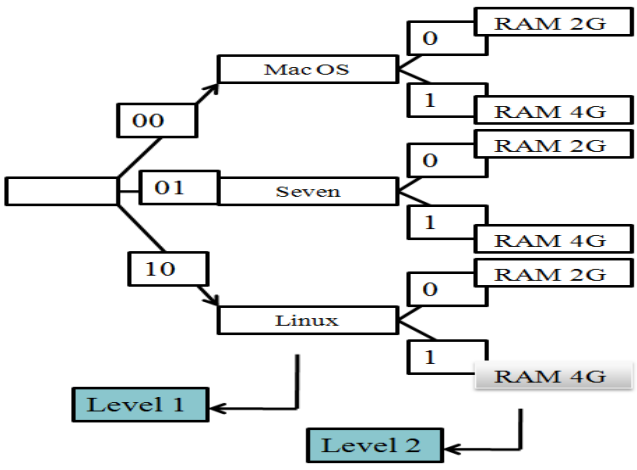


Fig. 1: An example of Resource-Tree (RT).

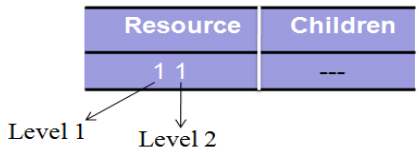


Fig. 2: The content of the highlighted field in Fig. 1.

Assume that the field in Fig. 2 has been stored in the place highlighted in Fig. 1. In *Resource* column, the number 11 has been stores. This means that this node possesses the resource level 1 (OS) and the resource level 2 (RAM). Considering the place where is stored in RT, it possesses Linux and RAM 4G. For better comprehension, you can look at Fig. 3 and Fig. 4. Fig 3 illustrates the assumed environment of our grid on a weighted tree structure. As seen in the figure, each node shares a resource or a combination of resources in the environment. How the resource information is stored inside some of the nodes is clearly seen in Fig. 4.

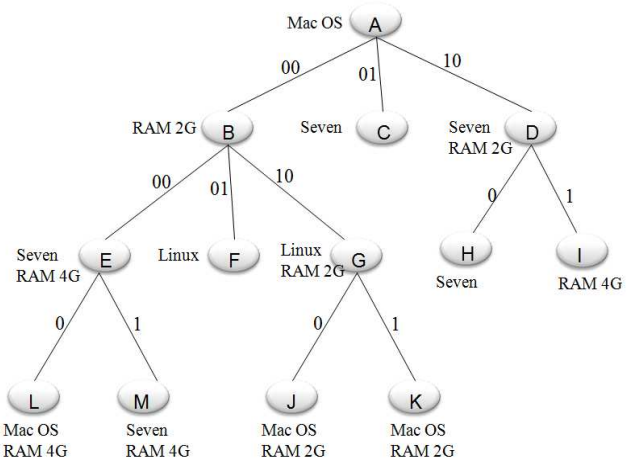


Fig. 3. An example of typical grid environment on a weighted tree.

Each part of Fig. 4 is explained subsequently. Just for simplification, O1, O2, O3, R1 and R2 will be used to refer to MacOS, Seven, Linux, RAM 2G and RAM 4G respectively. In Fig. 4(a), the RT stored in the nodes J and K are shown. As both nodes share the resources Mac OS and RAM 2G in the environment, they have similar RTs. Number 11 stored in *Resource* column means that in this place exists the information related to a combinational resource that possesses both OS and RAM, and they are Mac OS and RAM 2G with consideration of the place where they are stored. The mark "---" in the *Children* column indicates that these resources are the local resources of the node itself.

For another example, look at Fig. 4(b) related to the node G in Fig. 3. This node which receives information from its own children in addition to its own local information will store all this information in its RT as shown in Fig. 4(b). This node itself consists of O1 and R1 which will store the information of which as 11 in *Resource* column and mark "---" in *Children*

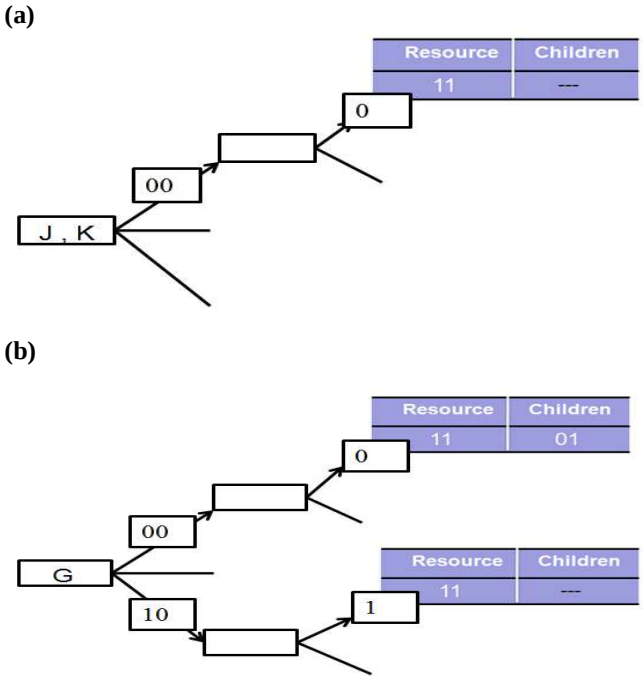


Fig. 4. The stored RTs in the (a) nodes J and K; (b) node G.

column, but will store the information related to children (O1 and R1 from both children) in the related place (number 11 in *Resource* column). Number 01 stored in *Children* column means that because this node has two children; therefore, it allocated at least 2 bits to each node, which is 0 and 1 as in Fig. 3. Since it receives similar information from both its children, the children's weight; that is 0 and 1 is written beside Children column (01).

It is pointed out that the information the method proposed here is stored in nodes distributary. This reduces the volume of the information stored in the nodes.

3.2 Multi-resource discovery

As seen in Fig. 5, there is a sample of the request form. The request form consists of two columns, *Location* and *Resource*. *Resource* Column resembles the column with the same title in RT, and its bits indicate the existence or non-existence of resources. The other column; that is, *Location* column, indicates a field to which referral will be made in every node in the course of the resource discovery.

Request	
Location	Resource
XX0	01

Fig. 5. A sample of Request form.

For example in Fig. 5, when the user needs a resource R1, the information related to R1 in RTs may be stored in each of three fields at the address of 000, 010 and 100 (Fig. 1). That is to say, for the applicant R1, the resource level 1; i.e., OS is not important, and only the second path ending in RAM is of importance.

As such, in *Location* column, sign XX (X means an unimportant state) is stored, and any field that receives this request searches for three fields at 000, 010 and 100.

In Fig. 6, a sample of the resource discovery is shown. As seen in this figure, a user needs the resources O1 and R1 simultaneously, and delivers a requested form as shown in this figure to the node D in the tree. Receiving this form, this node immediately refers to the same column in its RT using the position written in *Location* column, and compares *Resource* columns. But as it is seen, this position does not exist in the RT of node D. Thus, it passes the request to its parent node. Referring to a place in its RT, node A compares *Resource* column of the request with *Resource* column in the related place and finds a correspondence in the second line and sends the request to a child with weight 00. Node B, too, repeats this process, and delivers the request form to node G. Finding a correspondence in the related field and referring to *Children* column, node G notices this resource in children with edge 0 and 1 and sends the request to one of the nodes selectively (node J here). Finally, node J finds the requested resource for the user and reserves O1 and R1, and then sends a successful response to the origin node. As seen, the resource discovery method suggested in this paper is simple and does not meet any extra nodes.

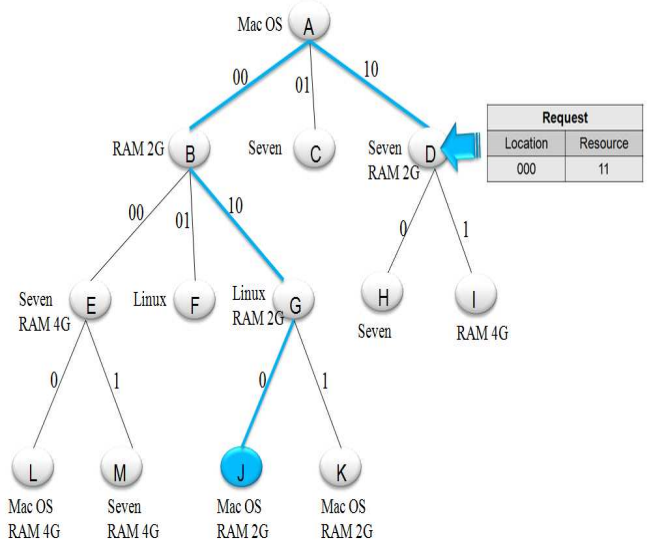


Fig. 6. An example of resource discovery in our method.

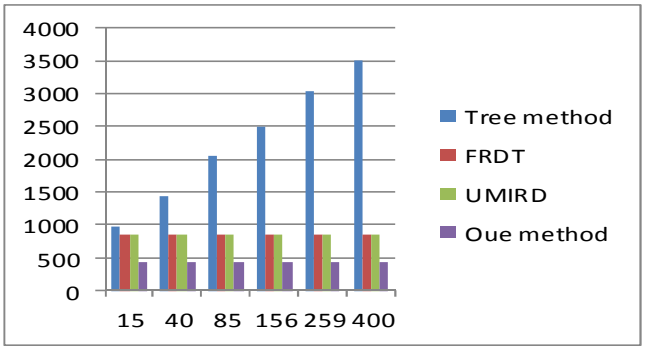
4 Simulation results

The simulations required for this work have been performed in MATLAB environment. The resources have been distributed randomly in this environment, and the requests, too, have been delivered to every tree node randomly. The height of the trees has been assumed 4 as in [5,6,19].

Since we did not find a method that can discover several resources in one format at the same time, we compared our method with other available methods with one resource. To compare multi-resources, we assumed that other methods discover the users' requested resources altogether in one place.

In the first simulations, we compared our method with "A resource discovery tree using bitmap for grids "[5] (which is called tree method), "Using Matrix indexes for Resource Discovery in Grid Environment" [8] (which is called UMIRD) and " FRDT: Footprint Resource Discovery Tree for grids "[6] methods. In Fig. 7, we supposed that the user requested different number of resources. In these tests, it is supposed that the 100 of the users, requested different number of resources. In our method, 100 requests will be sent but in other ones, for example for request six resources (Fig. 7(b)), 600 separate requests should be sent. As observed in the Fig. 7, the number of the nodes visited in our method is lower than other methods.

(a)



(b)

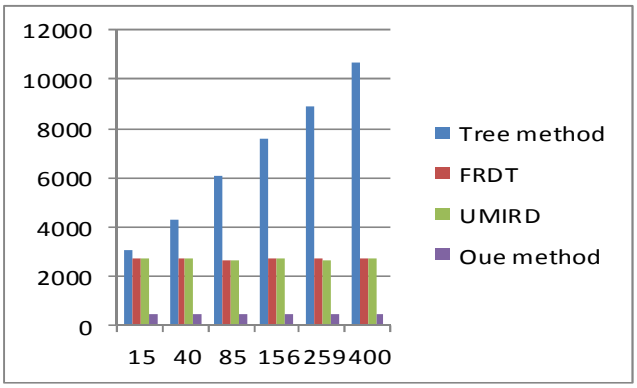
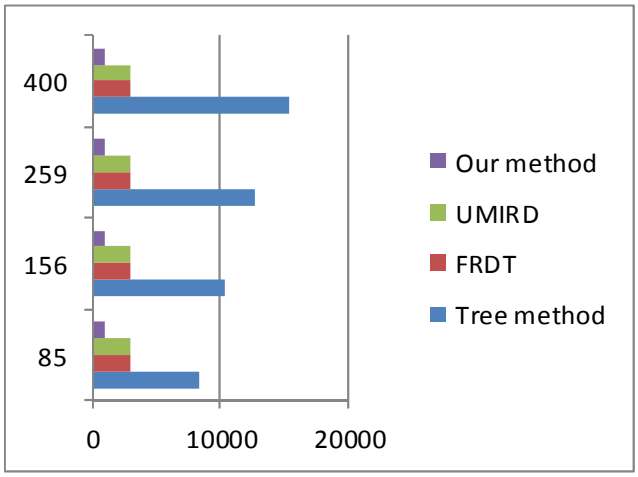


Fig. 7. The number of the nodes met by the users' requests during the resource discovery that the user requests; (a) two resources; (b) six resources.

In the next simulations, the traffic produced by the methods tree method, UMIRD, FRDT and our method upon resource discovery was compare, which is shown in Fig. 8. In these tests, it is supposed that the 300 of the users, requested different number of resources. As shown, our method is able to discover a desired number of resources for the user producing the least traffic and not referring to unnecessary nodes. Therefore, this method is more effective in the grid environment with many resources.

(a)



(b)

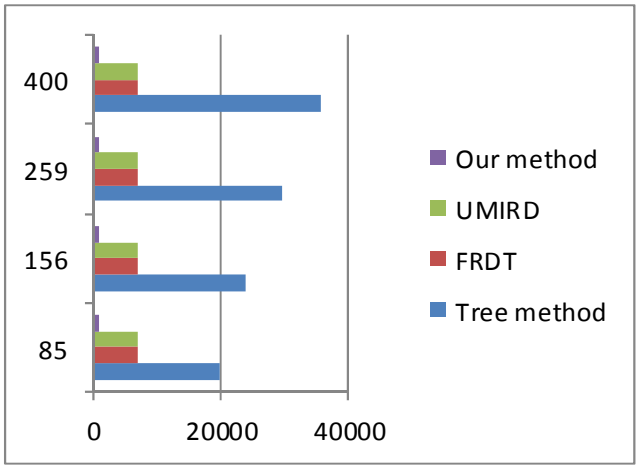


Fig. 8. The traffic produced by the different methods during the resource discovery for 300 users that each user requests: (a) three resources; (b) seven resources.

In the last tests, our method was compared with methods flooding-based, MMO [20-21], Tree method [5] and FRDT [6]. In this experiment, the mean of the number of met nodes was compared in different methods. It was assumed that any user would request only one resource (Fig. 9).

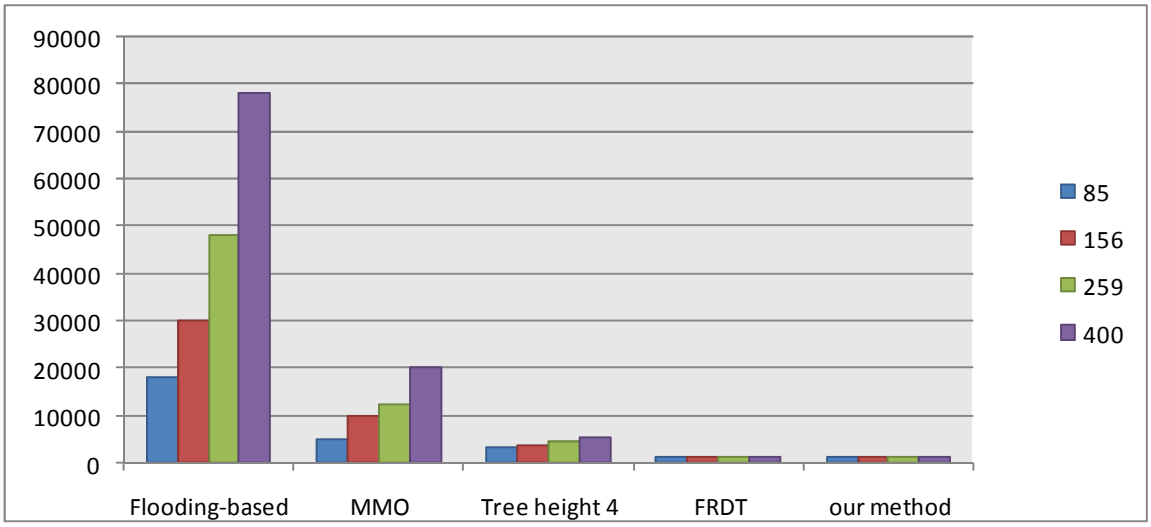


Fig. 9. The mean of the number of met nodes by different methods.

5 Conclusions and future work

As discussed earlier, we have proposed a method that is able to discover simultaneously the desired number of the resources for the user. In our proposed algorithm, the cost of the resource discovery is very low. The results of simulations indicate that our method is more efficient than other methods. In the future, the researchers will try to suggest a method which takes into account such factors as the cost, geographical distance, etc. for the resource discovery.

6 References

[1] I. Foster and C. Kesselman. "The Grid 2: Blueprint for a New Computing Infrastructure". Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003.

[2] A. A. Chien, B. Calder, S. Elbert, K. Bhatia, "Entropia: Architecture and performance of an enterprise desktop grid system", J. Parallel Distrib. Comput. (Elsevier), vol. 63, pp. 597-610,2003.

[3] F. Berman, et al., "Adaptive computing on the grid using AppLeS", TPDS, vol. 14, pp.369-382, 2003.

[4] M.O. Neary, S.P. Brydon, P. Kmiec, S. Rollins, P. Capello. "JavelinCC: Scalability issues in global computing". Journal of Future Gener. Comput. Syst. (Elsevier), Vol. 15, pp. 659-674, 1999.

[5] Chang, R-.S. and M-.S .Hu. "A resource discovery tree using bitmap for grids". Future Generation Computer Systems (Elsevier), vol. 26, pp. 29-37, 2010.

[6] L.M Khanli, and S. Kargar. "FRDT: Footprint Resource Discovery Tree for grids". Future Gener. Comput. Syst. (Elsevier), vol. 27, pp. 148–156, 2011.

[7] L.M Khanli, A. Kazemi Niari and S. Kargar. "An Efficient Resource Discovery Mechanism Based on Tree Structure". In the 16th International Symposium on Computer Science and Software Engineering (CSSE 2011), p. 48-53, 2011.

[8] Leyli Mohammad Khanli, Saeed Kargar, Ali Kazemi Niari. "Using Matrix indexes for Resource Discovery in Grid Environment". In the 2011 International Conference on Grid Computing and Applications (GCA'11), Las Vegas, Nevada, USA, pp. 38-43, 2011.

[9] R. Raman, M. Livny, M. Solomon, "Matchmaking: distributed resource management for high throughput computing", In the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7'98), pp. 140, 1998.

[10] Ye Zhu, Junzhou Luo, Teng Ma, "Dividing Grid Service Discovery into 2-stage matchmaking", ISPA 2004 (LNCS), vol. 3358, pp. 372–381, 2004.

[11] Sanya Tangpongprasit, Takahiro Katagiri, Hiroki Honda, Toshitsugu Yuba, "A time-to-live based reservation algorithm on fully decentralized Resource Discovery in Grid computing", Parallel Computing (Elsevier), vol. 31, pp. 529-543, 2005.

[12] Simone A. Ludwig, S.M.S. Reyhani, “Introduction of semantic matchmaking to Grid computing”, J. Parallel Distrib. Comput. (Elsevier), vol. 65, pp.1533 – 1541, 2005.

[13] Ami T.Choksi, Devesh Jinwala, “Improving Semantic Matching of Grid Resources Using refined Ontology with Complement Class”, Journal of AICIT, vol. 2, no. 5, pp.129-139, 2010.

[14] J. Li. and Son Vuong, “Grid resource discovery using semantic communities”. In the proceedings of the 4th International Conference on Grid and Cooperative Computing, Beijing, China, 2005.

[15] Juan Li, Son Vuong, “Semantic overlay network for Grid Resource Discovery”, In Grid Computing Workshop, 2005.

[16] Cheng Zhu, Zhong Liu, Weiming Zhang, Weidong Xiao, Zhenning Xu, Dongsheng Yang, “Decentralized Grid Resource Discovery based on Resource Information Community”, Journal of Grid Computing (Springer), vol. 2,no. 3, pp. 261-277,2004.

[17] Thamarai Selvi Somasundaram, R.A. Balachandar, Vijayakumar Kandasamy, Rajkumar Buyya, Rajagopalan Raman, N. Mohanram, S. Varun, “Semantic based Grid Resource Discovery and its integration with the Grid Service Broker”, In the proceedings of 14th International Conference on Advanced Computing & Communications (ADCOM 2006), pp. 84–89, 2006.

[18] J. Li, “Grid resource discovery based on semantically linked virtual organizations”. Future Gener. Comput. Syst. Vol. 26, pp. 361–373, 2010.

[19] Mastroianni, C., D. Talia and O. Versta. “Evaluating resource discovery protocols for hierarchical and super-peer grid information systems”. In the proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP’07, February 7– 9, pp. 147–154, 2007.

[20] Marzolla, M., M. Mordacchini and S. Orlando. “Resource discovery in a dynamic environment”. In the proceedings of the 16th International Workshop on Database and Expert Systems Applications, DEXA’05, September 3–7, pp. 356–360, 2005.

[21] Marzolla, M., M. Mordacchini and S. Orlando. “Peer-to-peer systems for discovering resources in a dynamic grid”. Parallel Comput. Vol. 33, pp. 339–358, 2007.