Journal of Theoretical and Applied Information Technology
15 August 2012. Vol. 42 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DYNAMIC TTL BASED ALGORITHM FOR HIERARCHICAL RESOURCE DISCOVERY MODEL IN GRID

[1]KAMRAN ZAMANIFAR, [2]HOSSEIN ABOOTALEBIAN, [3]LADAN MALAZIZI
[1,2,3]Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad-Iran
E-mail: [1]zamanifar@eng.ui.ac.ir, [2]hossein.abootalebian@gmail.com, [3]lmalazizi@pco.iaun.ac.ir

ABSTRACT

Today, Grid is used as a powerful infrastructure for resource sharing. One of the main services in the Grid is the resource discovery service. Several models and methods have been proposed to manage and search of resources. The central model has problems such as single point of failure and heavy load. There are newer models in order to solve the mentioned problems, but there is also the problem of sending anomalous query messages among these models.

In this paper, resource discovery tree which is constructed based on hierarchical model has been utilized. In the first place, by considering coefficients for each level in the tree, the managers are provided with information. Afterward, by using this information, time to live of discovery messages are dynamically determined. By doing so, the forwarding of these messages is limited so that the traffic imposed by resource discovery is alleviated and its performance is enhanced.

Eventually, to evaluate, the simulation of the presented method through this paper and other methods were used. The results of the accomplished evaluation showed that although the proposed method uses small number of update messages and spends a little time to process the information of the resources, it significantly decreased the traffic resulted from the transferring of the messages for resource discovery and also increased speed.

Keywords: *Grid, Hierarchical Model, Resource Discovery, Time to Live*

1. INTRODUCTION

Considering the rapid growth of technology, human being's need to perform the complicated calculations, storage space, and advanced equipment will increase day by day. In the 1998, Grid was emerged to build an infrastructure in order to resolve these needs [12]. Analogous to power Grid, the computing Grid is perceived as a Grid of distributed computers that provides a transparent and pervasive computing infrastructure in which computing capability is delivered over the Internet and can be used as a utility [18]. The Globus project (Argonne National Laboratory, USA) defines Grid as "an infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations". An important service of grid computing is resource discovery. Users are not interested in where resources actually are. Just by giving a description about resources they desire, the resource discovery mechanism will find a set of resources that match the user's description if there exists one [18]. An effective Resource Management is required for the provisioning and sharing of resources without undermining the autonomy of their environments and independence of geographical locations [16].

There are so many methods to search the resources on the Grid which each method performs its own job based on a model. Central, Hierarchical, Peer-to-Peer, and Super-Peer are considered the main models of resource management in the Grid. A centralized approach can easily manage all the resources. However, when the types and amounts of resources become large, all queries and other operations have to go through the centralized server. It will be the bottleneck of the system and when it fails, the whole system ceases to function. Therefore, the centralized architecture is not suitable for resource discovery in a distributed grid environment [18]. The hierarchical model shows good performance for small and medium-sized Grids, and owing the client-server structure is of highly important ones and nowadays, they are widely used [3]. Super-Peer and Peer-to-Peer models are suitable for very large-sized Grids.

One of the problems in the process of finding of the user's desired sources is the traffic resulting the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

sending of the query message. So, some methods are required in order to reduce the traffic. To solve this problem, a method is presented based on the hierarchical model through this paper. Our method which will be presented in detail, used Dynamic Time to Live or Dynamic TTL to prune the unnecessary traffic. This is done by calculating of children's weight. The allocated weight shows that the existence of the resource in a node preferred to that resource in all its children. So, when the weight of nodes to be calculated and gave to higher levels, they are able to make more accurate decision about the query message TTL in order to transfer fewer message to the children for discovering the related resource. In the following, we discuss about the similar works done in this area, explaining the present method, giving examples, and evaluating and considering the results.

2. RELATED WORKS

In Grid, when a user needs a resource, it sends its request. In order to find the user's desired resource, there are so many methods and models. Some methods do their task based on the Centralized model [8]-[15]. In Centralized model, management and resource search is performed by a central manager. The disadvantage of the model is creating the point of failure and Bottleneck in resource manager. Nowadays, many methods have been implemented based on the Hierarchical model [18]. Some methods have also been implemented based on Peer to Peer [20] and Super-Peer [3]-[4] models.

Yangcai Tao et al [24], have used a tree structure to search the resources. The Index Servers, which are in one level, are classified in one domain. During discovery of resources, if an Index Server hasn't a resource, it will inform the index servers of its domain.

Ramos and de Melo [21], have used the Master-Slave structure to do their tasks. Slave task is collecting the data from the machines, and Master is responsible for updating the resource information.

Xue-Sheng Qi [23], keeps the resources information in a table. The stored resources are registered in the table and used to discover the user's desired resource. When the user is going to find a resource, first he considers this table. If the mentioned table not to exist, he will create it. Though Forward Path, when a node receives the request, it considers that whether the related resource is present in local resources or no?. If it presents, the relevant resource will be reserved and add to this table then the request will send to the next node. Through Backward Path, the request to be sent toward the resource requestor from the same path it has been come. The user considers the table by which he finds his own desired resource.

Ruay-Shiung Chang, Min-Shuo Hu [18], proposed a tree architecture for resource discovery. All resources and queries are transformed into a bitmap index representation. Each leaf node in the tree will store the information about its local resources. Each Index Server, will store the information about its local resources and the information about its children. The index in each Index Server is the bitmap of bitwise OR of all its children nodes. by AND operation, check whether a node has the resources to satisfy user's requests.

L. Mohammad Khanli, S. Kargar [11], use of a weighted tree and bitmap where the number of bit positions is proportional to the number of attributes for each resource, for resource discovery in Grid.

Difference between our algorithm and other algorithm is using of coefficient for each tree level. By this, we can calculate efficient TTL for sending queries.

3. OUR METHOD

As stated, one of the most important Grid services is resource management and discovery service. Also explained that the most used models in the implementation of this service are centralized, hierarchical, Peer-to-Peer, and Super-Peer models. The Centralized model leaves the resource management to one resource manager. So, all the Grid members transfer their information to the mentioned manager and when a member needs a resource, transfer his/her request to the manager. So, how to respond to all the requests and the way of allocating all resources is the responsibility of this manager. Simplicity is a positive feature of this method, but, there are also problems of which can be noted as point of failure and high terrific in resource management. So, this model is not a suitable one for an environment which is based on distribution.

Other models instead of resource management as Centralized, implement it as distributed. These models employ several managers who are distributed in the entire of the Grid and connected to each other instead of one manager. The way of connecting the managers to each other is based on the different models. For example, the connection

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of the managers to each other as a tree is based on the hierarchical model.

As stated earlier, the hierarchical model indicates a high performance in small and medium-sized Grids, and nowadays, they are widely used. Super-Peer and Peer-to-Peer models are suitable for the Grids with a huge size [4]. In this paper, we use hierarchical model as a tree structure. We intend to reduce the number of queries toward the tree according to available resources and the request which issues for holding them using Dynamic TTL.

In the present structure, the tree nodes indicate the Grid nodes. Non-leaf Grid nodes present a service as Index Service. So, they are called Index Server. Grid nodes can be connected to each other with any physical structure. The presented tree structure is created based on the logical connections such as IP Address. So, each node in the tree knows its father and children. Non-leaf nodes or Index Servers are not only aware of their resources situation but also of all or a number of their children. The awareness amount of an index Server from the children resources is based on the method. But if an Index Server wants to maintain all the information related to its children, it is not a suitable, due to the storage space and its processing and the information security.

Ordinarily, when a tree structure is used to search the resources in the Grid, when a request is made, the local resources will be reviewed to respond to the request, if local resources can respond to the request, the resource will be allocated. If local resources are unable to respond to the request and the node which the request derived from, not to be the leaf node, the request is searched through the children. If children can't respond to the request and resource requester not to be the tree root, the request is transferred toward the father. This process is recursively repeated until either a resource to be found or get to the root of the tree and also not to be found any resource in the root. The above-discussed cases are the basic of the work. Different methods concentrate on the way of transferring the request to the children and reduction of the number of query messages and updating through the tree. In algorithm presented in this paper, the Dynamic TTL is used to select the children. We intend to, avoid scattering unlimited number of requests through the tree during transferring the search to children, based on the information which we have about them. For example, assume that the node itself has not the requested resource, but one of its children possess it. If the request transfers to the children with no restriction, the one who has the resource, receives the request and respond it, but the children who have not the resource try to transfer the request to their own children and it continues in order to reach a node which has the resource or reach the tree leaves, so transferring these messages has not been needed. Therefore, by preparing little information about the children according to the algorithm, a TTL is determined for the request in order that the message goes equal to the TTL. By this, inept sending the query message in the tree is prevented. So, in this example, before transferring the message to the children, TTL will be equal to one. Based on the assumptions of the example, the response to a request is made in the first level, and because of the TTL which is one, the request has no forwarding more than one level in the tree. Furthermore, assume that during transferring the request to children according to our information, we understand that the request is available through the children, but more than one level must go forward to reach the resource. Assume that the resource is available in two lower levels in a child. So, we put TTL as 2 and transfer the request to all the children. When the request is obtained by the children, if a child does not have the resource and by his information realizes that with TTL 2 cannot respond to this request through its own children, it will not transfer the request for its own children. So another reduction is gained in transferring the query messages. Based on the above mentioned issues, it is clear that in order to implement the algorithm, it is needed to know that if we want to search the request through children, the closest child who has the resource is located after how many levels?. Moreover, if a request is transferred for the resource and request more than one number from that resource, what number of the children has this resource and forward to how many levels in order to find this number of children?.

The details of our method to solve the aforementioned problems will be discussed as follows.

3.1 Algorithm of Making Resource Discovery Tree

It should be noted that for each resource, the steps described below, is performed. Each node must provide information about itself and its children to father for each resource. So for the exchange of information through the tree, we start from the tree leaves. If a leaf has the resource, it will transfer its own coefficient, which is shown in

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TABLE 1. Level Coefficient

h is tree height, *m* is tree degree and *a* is constant number

Level	Coefficient
0=Root	$(m+1)(m^2+1)\dots(m^{h-1}+1)a$
1	$(m^2+1)\dots(m^{h-1}+1)a$
....	...
h-3	$(m^{h-2}+1)(m^{h-1}+1)a$
h-2	$(m^{h-1}+1)a$
h-1	a

Table.1, for the father, otherwise use the 0, as the information of itself about that resource. The information of a node about resource *r*, called I_r. Father waits to receiving the message from all the children. When the father receives the number related to all children, add them with each other and if he himself has that resource, add his coefficient with it and keep result in itself as the information from that resource. Then, transfer a copy of it for his father. It continues to reach the root. The example is present in fig.1. By preparing this information, a node can understand the number of free nodes that its children have in each level from each resource.

3.2 How to handle Requests

When, for example, resource *r*, is requested at a node, it must be investigated whether local resources are responsive to the request or not. To do this, I_r of this node is divided into the coefficient of this level. If the integer part of the division will be 1, we understand that the related resource of the current node is available. So, the I_r of this node is reduced as the coefficient of this level. If the number of the requests is one number, resource will be reserved, and based on the present address in the message of the request, will inform the requester, so the search will be terminated in this point. If the integer part of the division results to be 0, i.e. the resource *r* not to be present in this node, or the number of the requested resources to be more than one, the I_r of the node must be divided into the coefficient of the next level. The integer part of the resulted number indicates the number of the children in the next level which have the related resource. If the remaining number of the requested resources are smaller or equal to number of the children which possess this resource in the next level, TTL will equal 1 and the message of request with TTL=1 transfer to the children. Otherwise, the number of the children in the next level, who has the resource, multiplies in the coefficient of their level and the resulted amount reduces the I_r of the node. Then, the TTL of the related message increases one unit. Now the outcome number of the subtraction divide into the children coefficient of the following two next levels, so it is considered that whether the number of the resources in next level and next two levels are the responsive of the request or no? So, TTL of message is also added one unit. This is repeated until the present children in the tree leaves are also reviewed. If the number of the requested resources to be more than the total existing resources of the sub tree, the request transfers to father. Otherwise, the message of request with the related TTL transfers to the children after determining the corresponding TTL with the number of the requested resources. The child who receives this message, he firstly considers that whether the requested resource is present through itself or no? If it exists, he reserves the resource and informs the resource requester, and then he subtracts one unit of TTL of the message. If TTL equals 0, or the coefficient of the last level which could reach it with the present TTL, it is bigger than the I_r of this child i.e. reaching a child with this TTL to be impossible which possess this resource, transferring the message will be terminated. Otherwise, the message will be transferred to the children and continues in this way. When the requester receives the reservation message, informs the message sender its acceptance or its rejection. If the resource to be accepted, it will be put in requester's hand, otherwise, it will be free. When the requester's work terminates with the resource, the resource will be free. During reserving or freeing the resource, the tree must be updated which will be discussed in the following steps. Through the presented algorithm, if the requester's number of the requested resources equals N, at least, N answers will give to it.

Journal of Theoretical and Applied Information Technology
15 August 2012. Vol. 42 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

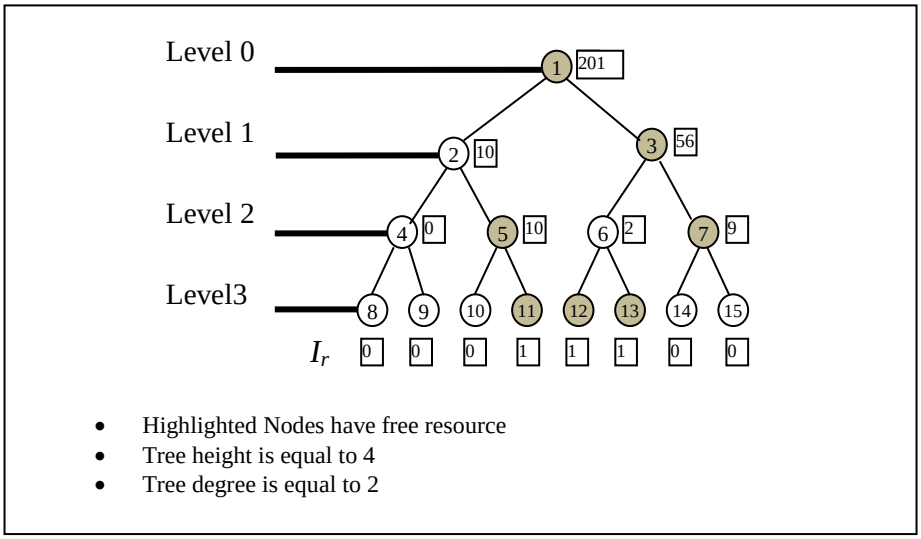

Figure. 1: Example.

Now, look at an example to explain the above. Suppose that node No.2 in fig.1 want two number of resource *r*. Because the coefficient of level one equals 45, and number 45 is more than I_r of this node, i.e. 10, we understand that the related resource is not available in the local node. So, the coefficient of the next level, i.e. 9, is considered. Now, 10 divides into 9, its result will be 1.1. Because the integer part of this number equals 1, we understand that, there is one resource of requested resource kind, in the next level. So, if 9 subtract 10, it remains 1, now the coefficient of next two levels divides into 1, i.e. 1 divides into 1. Its result equals 1, i.e. in next two levels there is one number of this resource. So in order to obtain 2 numbers from this resource, it is needed to go forward two levels among the children. So, we put TTL as 2 and the message which carries the request as well as TTL will be transferred to children. The children in level 2 receive this message. The child having No.4 considers that the TTL related to the level in which he presents, equals 9, and 9 is bigger than his number that is 0, so he understands that hasn't this resource. Now, TTL is subtracted one unit, TTL equals 1. The coefficient of the one next level equals 1 which is more than the I_r of this node which is 0. So, he understands that among his children, he has no reach to the resource with TTL equals 1. So transferring the message will be terminated. The child No.5 also receives the message. The number of the second level equals 9. The number of this node equals 10. 10 will be divided into 9. Its result equals 1.1. So, he himself has this resource. He reserves the resource and will inform the requestor. Now, he subtracts 9 from 10, its result equals 1. So, considering that the coefficient of the next level is 1, he understands that he can find this resource among his children. In this case, he reduces TTL and because TTL doesn't equal to 0, the message will be transferred to his children. Through this method, the child NO.11 also receives the message and reserves the resource and informs the requestor. But the node No.10 understands that he hasn't this resource. So, due that the TTL equals 0 and being leaf node, transferring will be terminated. In this case, at least two answers will give to the requestor.

3.3 UPDATE

When the resource is reserved, the I_r is subtracted equal to the coefficient of the node level, and the result will be transferred for the father. The father also does updating and transfers its result for his father. It continues to reach the root. Moreover, when the resource to be free, it will be add equals to the coefficient of node level to I_r of the node and keep updating to root. So, with fewer numbers of messages, updating will be done.

4. EVALUATION

Simulation has been performed via Arena Software (produced by Rockwell Company). Through this simulation, we are going to calculate and compare the amount of traffic resulting of the search of a specific resource for the delivered algorithm in the current paper, Flooding-based and the delivered approach in [18].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

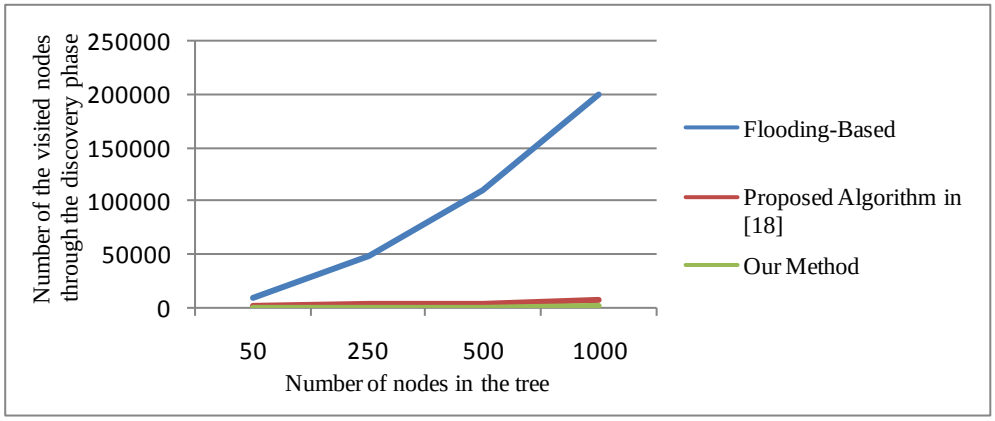

Figure. 2: Number of the visited nodes through the discovery phase for 100 queries

In order to do so, the resources are randomly distributed among the tree nodes. Through the first experiment, we intend to send 100 queries among the tree nodes, and count the number of the visited nodes by any of the algorithms, and compare the obtained results by each other. The number of nodes in the tree and related results of this experiment are shown in fig.2. These results show that in Flooding-based approach a large number of nodes are met due to anomalous sending of queries. Through the presented approach in [18], because of adding more information to the nodes, better decision is taken in sending the query to the other nodes. So, the number of the queries decreases. Using the information which is added to the nodes by the presented algorithm in this paper, we are still able to reduce the number of queries. In the second experiment, we calculate the number of the nodes in which updating to be occurred after finding the desired resource. In order to do updating by the presented algorithm in [18], it is sufficient that each node sends updating toward father. So, each of the updates goes toward the height of the tree in, until, reach the root. Due that the Flooding–based approach has no information on its neighbors and because of sending the query to all its neighbors, so it has no need to update. The presented method in this paper, due to discovery of closer resources to the requestor, the path which any updates passes is shorter. The results of this experiment are shown in fig.3. Eventually, the total number of the queries and updates resulting of these 100 requests for each three methods is shown in fig.4. Although, the Flooding-Based approach does no updating, but on the whole, due to anomalous sending of query, it has no efficiency. The presented method in [18], still indicates better proficiency towards Flooding-Based approach with regard to updating.

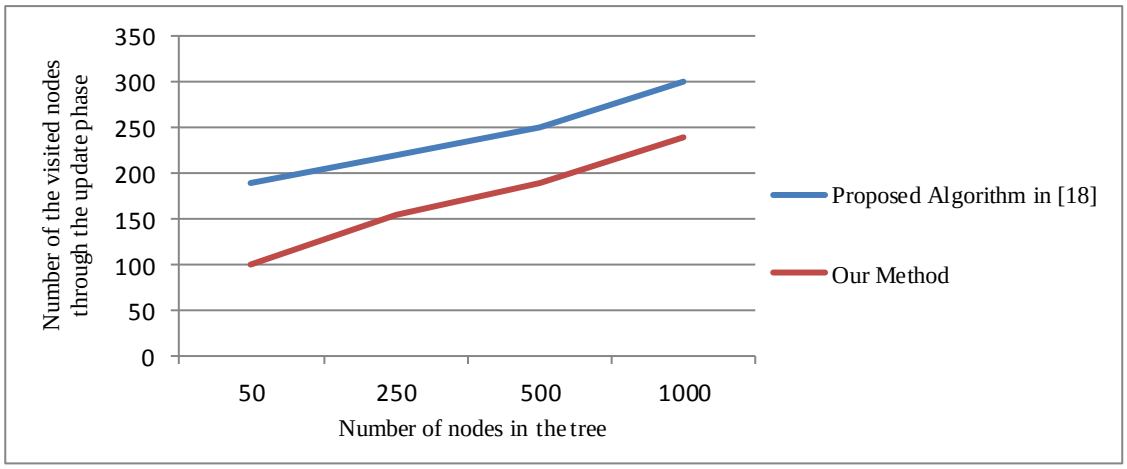

Figure. 3: Number of the visited nodes through the update phase for 100 queries

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

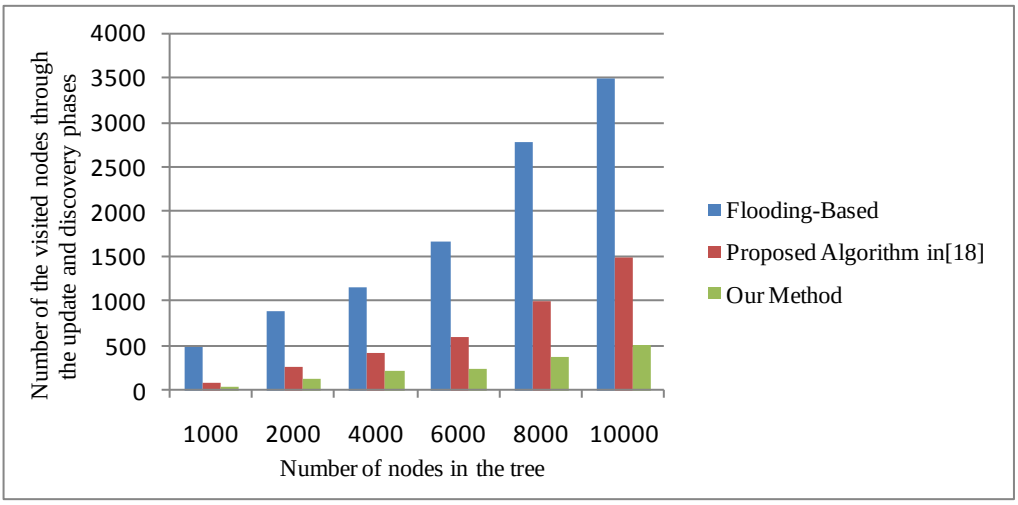

Figure. 4: Number of the visited nodes through the update and discovery phases for 100 queries

The presented method in this paper is increase the efficiency of the presented method in [18], in an significant amount due to more cautious sending of query and selecting the closer children. Considering the results of evaluation, the presented method in this paper creates less traffic on queries to find the user's appropriate resource in order to deter the exponential growth of the queries. Furthermore, in comparison with other algorithms, few answers are sent toward requestor and deter the traffic on the requestor of the resource.

5. CONCLUSION AND FUTURE WORKS

The presented method is concentrated on reducing the number of messages resulting of finding resources. To achieve this goal, some coefficients are considered for the tree levels, then using these coefficients, some information was provided for the fathers of each node and aiding this information, the TTL of the query message is determined and so the progression rate of the query message is limited. The results of the simulation indicate that during searching the resources, the visited nodes are less than the other methods by our method.

In future works, the increase of efficiency should be concentrated. One of the issues is that if the tree levels to be increased, the coefficients will be very large and working with them is so difficult. Therefore, determining more efficient coefficients is of paramount importance. The cost and quality of resources are also of other parameters which can be considered in order to determine the TTL of the query message.

Moreover, we can continue our work based on Super-Peer model which is suitable for the Grids with high large size.

REFERENCES

[1] A.R. Bharambe, M. Agrawal, S. Seshan, Mercury: Supporting scalable multiattribute range queries, in: Proc. of ACM SIGCOMM, 2004, pp. 353–366.

[2] B Jacob, M Brown, K Fukui, and N Trivedi, Introduction to grid computing, 1st ed.: IBM Red book, 2005.

[3] C. Mastroianni, D. Talia, O. Verta, A super-peer model for resource discovery services in large-scale grids, Future Gener. Comput. Syst. 21 (8) (2005) 1235–1248.

[4] C. Mastroianni, D. Talia, O. Versta, Evaluating resource discovery protocols for hierarchical and super-peer grid information systems, in: Proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP'07, February 7_9, 2007, pp. 147_154.

[5] E Huedo, R S Montero, and I M Llorente, "A recursive architecture for hierarchical grid resource management," Future Generation Comp. Syst, vol. 25, no. 4, pp. 401-405, February 2009.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[6] F Magoulès and C-H Li, Grid resource management: toward virtual and services compliant grid computing, 1st ed. Boca Raton: CRC Press, 2009.

[7] H Shen, "A P2P-based intelligent resource discovery mechanism in Internet-based distributed systems," J. Parallel Distrib. Comput, vol. 69, no. 2, pp. 197–209, 2009.

[8] I. Foster, C. Kesselman, Globus: A metacomputing infrastructure toolkit, Int.J.High Perform. Comput. Appl. 2 (1997) 115–128.

[9] I. Foster, C. Kesselman, and S Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," International Journal of Supercomputer Applications, vol. 15, no. 3, pp. 200–222, 2001.

[10] K. Czajkowski, S. Fitzgerald, I. Forster, C. Kesselman, Grid information services for distributed resource sharing, in: Proc. 10th IEEE International Symposium on High-Performance Distributed Computing, HPDC-10, August, 2001, pp. 181_194.

[11] L.M. Khanli and S. Kargar, "FRDT: Footprint Resource Discovery Tree for grids", presented at Future Generation Comp. Syst., 2011, pp.148-156.

[12] M. Li and M. Baker, The Grid: Core Technologies, 1st ed. Chichester, West Sussex PO19 8SQ, England: John Wiley and Sons, 2005.

[13] M. Marzolla, M. Mordacchini, S. Orlando, Resource discovery in a dynamic environment, in: Proceedings of the 16th International Workshop on Database and Expert Systems Applications, DEXA'05, September 3_7, 2005, pp. 356_360.

[14] M. Marzolla, M. Mordacchini, S. Orlando, Peer-to-peer systems for discovering resources in a dynamic grid, Parallel Computing 33 (4_5) (2007) 339_358.

[15] M.O. Neary, S.P. Brydon, P. Kmiec, S. Rollins, P. Capello, JavelinCC: Scalability issues in global computing, Future Gener. Comput. Syst. J. 15 (5–6) (1999) 659–674.

[16] M. Siddiqui, T. Fahringer Series: Lecture Notes in Computer Science / Theoretical Computer Science and General Issues, 5951, ISBN: 978-3-642-11578-3.

[17] R. Buyaa, K. Bubendorfer, Market-Oriented Grid And Utility Computing, Wiley, ISBN 978-0-470-28768-2, 2009

[18] R.-S. Chang, M.-S. Hu, A resource discovery tree using bitmap for grids, Future Gener. Comput. Syst. 26 (2010) 29–37.

[19] S. Tangpongprasit, T. Katagiri, H. Honda, T. Yuba, A time-to-live based reservation algorithm on fully decentralized resource discovery in grid computing, Parallel Computing 31 (6) (2005) 529_543.

[20] Simon G.M. Koo, Karthik Kannan, C.S.George Lee, On neighbor selection strategy in hybrid Peer-to-Peer networks, Future Gener. Comput. Syst. 22 (2006) 732–741.

[21] T.G. Ramos, A.C.M.A. de Melo, An extensible resource discovery mechanism for grid computing environments, in: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid, CCGRID'06, May 16–19, 2006, pp. 115–122.

[22] V Vassilios and E Pitoura, "On the performance of flooding-based resource discovery," IEEE Trans. Parallel Distrib. Syst., vol. 17, no. 11, pp. 1242-1252, 2006.

[23] X.S. Qi, K.L. Li, F.J. Yao, A time-to-live based multi-resource reservation algorithm on resource discovery in Grid environment, in: Proceedings of the 2006 1st International Symposium on Pervasive Computing and Applications, August 3_5, 2006, pp. 189_193.

[24] Y. Tao, H. Jin, X. Shi, L. Qi, GNSD: A novel service discovery mechanism for grid environment, in: Proceedings of the International Conference on Next Generation Web Services Practices, NWeSP'06, September 25_28, 2006, pp.17_26.