RDVBT: Resource Distance Vector Binary Tree Algorithm for Resource Discovery in Grid

SeyedElyar Hashemseresht*, Ali Asghar Pourhaji kazem**
* Department of Computer engineering, Tabriz branch, Islamic Azad University, Tabriz, Iran
** Department of Computer engineering, Tabriz branch, Islamic Azad University, Tabriz, Iran

Article Info

Article history:

Received May 16, 2012
Revised Jun 6, 2012
Accepted Jun 16, 2012

Keyword:

Grid Computing
Resource discovery
RDV

ABSTRACT

Nowadays, with the increasing variety of computer systems, resource discovery in the Grid environment has been very important due to their applications; thus, offering optimal and dynamic algorithms for discovering resources in which users need a short period is an important task in grid environments.One of the methods used in resource discovery in grid is to use routing tables RDV (resource distance vector) in which the resources are based on certain criteria clustering and the clusters form a graph. In this way, some information about the resources is stored in RDV tables. Due to the environmental cycle in the graph, there are some problems; for example there are multiple paths to resources, most of which are repeated. Also, in large environments, due to the existence of many neighbors, updating the graph is time-consuming. In this paper, the structure of RDV was presented as a binary tree and these two methods (RDV graph-algorithm and RDVBT) were compared. Simulation results showed that, as a result of converting the structure to a binary tree, much better results were obtained for routing time, table updating time and number of successful requests; also the number of unsuccessful requests was reduced.

Corresponding Author:

Seyed Elyar Hashemseresht,
Departement of Computer Engineering,
Islamic Azad University of Tabriz
Pasdaran highway, Tabriz, Iran, Islamic Azad University
Email: e.hashemseresht@iaut.ac.ir

1. INTRODUCTION

Recent advances in network bandwidth and speed of communication make humans move toward distributed computing environments in order to solve complex problems much faster. Grid computing is different from formal distributed computing and cluster computing; it is based on large scale resource sharing and new applications over a wide area network connections like the internet [1]. The mechanism provided by the grid structure should be available to find a suitable resource for a request. Therefore, one of the important capabilities a grid structure needs to support is a resource discovery mechanism. Finding a special resource in traditional computing systems is easy because the number of shared resources is few and all resources are under central control. But, by deploying grid systems, finding appropriate resources for the incoming requests in a short time is important in the resource discovery mechanism. Also, in a grid environment, there are specific factors that make the resource discovery problem difficult to solve. These factors are the number of resources, different ownership, resource failure, heterogeneity of resources and so on. An efficient resource discovery algorithm should consider above factors. Moreover, some other factors such as being online or offline because of the dynamicity of grid structure are important.

There are many approaches for resource discovery such as random-based or tree using bitmap [2]. Another approach is resource discovery graph using resource distance vector routing table (RDV) [3], in which resources are based on certain criteria clustering and the clusters form a graph. Some information on the resources is stored in RDV tables. Due to the environmental cycle in the graph, there are some problems such as multiple paths to resources, most of which are repeated. Also, in large environments, due to the existence of many neighbors, updating the graph is time-consuming. In this paper, a binary tree was used instead of a graph for resource discovery. Two distance vectors were applied for exploring available resources in nodes; one distance vector for local resource and the other one for the resources available in its neighbors and neighbor to neighbor nodes. The remainder of this paper is organized as follows: in Section 2, the related works are described and, in Section 3, the problem is defined. Resource distance vector mechanism is explained by a graph and RDVBT in Section 4. Section 5 shows the simulation and compares the experimental results of the method with those of the previous ones; conclusions are presented in Section 6.

2. RELATED WORKS

When users want to execute jobs in a grid environment, an algorithm should find the suitable resources for their requests. Because of the dynamicity of grid environments, resources constantly change and new resources and services are added to or are leaving the system. There are different methods for resource discovery in grid environment. One of the resource discovery methods is Monitoring and Discovery Service (MDS). Monitoring is responsible for resources and discovery has to detect appropriate resources.

Another method for resource discovery problems is centralized resource discovery approaches [5-10], the problem of which is that they become the bottleneck.

Sanya tangpongprasit et al. [11] proposed an algorithm which used the reservation mechanism for finding suitable resources in a grid environment. In the forward path, if there were any resources, they would be saved and reserved in the backward path, one of which would be selected and added to the request (if more than one resource was reserved). The algorithm used the experienced-based random rule to decide the node to forward the request.

Chang and Hu [12] proposed a resource discovery tree using bitmap for grids which used two bitmaps called ‘‘index bitmap’’ to register the information about its children nodes which existed in the nodes with children (non-leaf nodes) and ‘‘local resource bitmap’’ to register information about the local resources of nodes. In this method, the users’ request became AND with the local resource bitmap, at first, and if there was no local resource in the node, it would become AND with the index bitmap. If the result of the AND operation was zero, it meant that there were no resource in children and the request would be sent to the father node; if the result was a nonzero number, it meant that there were at least one resource in children so the request would be forwarded to all children until reaching the target node.

In [3, 4], Juan Li proposed a resource distance vector graph algorithm using routing table for grid. In this method, each node in the overlapping network could include a graph, every node of which had its own routing table. These routing tables were composed of two parts: the first part for holding the information of local resources and the other one for holding the resource that were available in its neighbors. In fact, information of the tables was vector numbers and each number had the minimum distance from the resource. In Section 4, this method will be described by an instance.

3. DEFINING THE PROBLEM

Resource discovery algorithms use a random-based approach for finding resources in grid environment; because of that the requests are sent to unnecessary paths, the system efficiency decreases and makes high traffic.

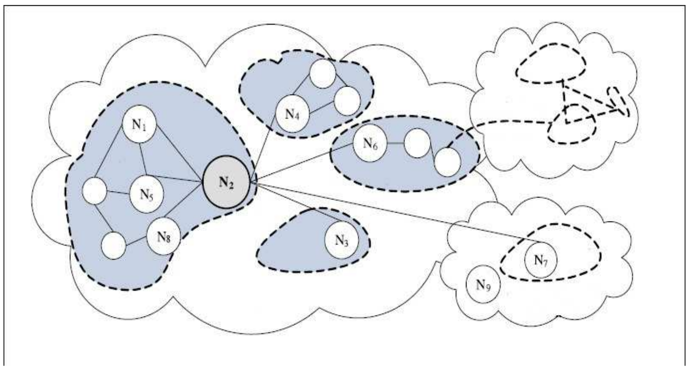


Figure 1.Overlapping network topology

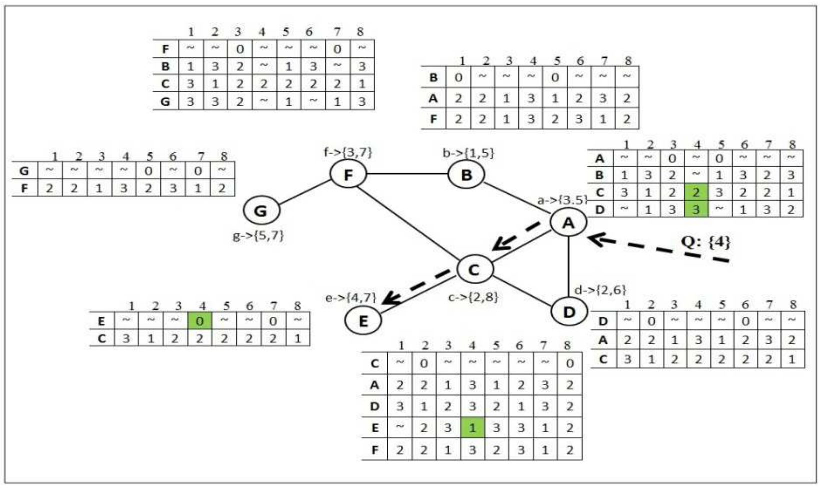
However, the resource distance vector (RDV) graph algorithm [3,4] improves many defects of the previous methods like heavy traffic and cost of update. In this method, each node in the overlapping network can include a graph with its own routing table which is shows in Figure 1.

In this way, the nodes have a routing table composed of two parts: the first part for holding the information of local resources and the other one for holding the resource that are available in its neighbors. A number of 8 different resources were considered for every node, each of which can be CPU, RAM, an operating system and so on. In fact, the information of the tables was as a vector of numbers and each number demonstrated the minimum distance to the resource. When the information of one node is sent to another, it is added with one unit. A radius is used to limit the number of steps of sent data. When a request is sent to one node, its local vector is checked and, if the resource does not exist, its neighbor vector is checked. If the resource exists in multiple nodes, the request is sent to the nearest one.

4. THE PROPOSED METHOD: RDVBT

The pervious method have some problems such as: 1- If the node has many neighbors, its routing table has many rows and it is time consuming to check them, find resources for requests and update them. 2- The number of updates of nodes' routing table for check of their resources online or offline is large, especially when the grid environment is very large. 3- There are multiple paths to resources, most part of which are the same and are repeated many times because of the environmental cycle in the graph.

For the reasons mentioned above, a binary tree was used here instead of a graph in order to decrease the time of finding resources and updating routing tables because of being online or offline. In this method, the routing table of each node had only three rows and these rows was checked because, in a binary tree, each node had two children; but, in a graph for example, if one node had five neighbors, the routing table had six rows and, when the grid environments were very large and each node of graph had many neighbors, so the routing table of nodes was huge and checking it for finding resources for the requests and updating them was time consuming. Both of these methods were shown by an example. Figure 2 shows an example of RDV tables in graph and Figure 3 shows the RDVBT method. The algorithm that was used to find resources for any request in the RDVBT method is shown in Figure 4. In both methods, radius was used for limiting the number of steps of the sent data.



	1	2	3	4	5	6	7	8
F	~	~	0	~	~	~	0	~
B	1	3	2	~	1	3	~	3
C	3	1	2	2	2	2	2	1
G	3	3	2	~	1	~	1	3

	1	2	3	4	5	6	7	8
B	0	~	~	~	0	~	~	~
A	2	2	1	3	1	2	3	2
F	2	2	1	3	2	3	1	2

	1	2	3	4	5	6	7	8
G	~	~	~	~	0	~	0	~
F	2	2	1	3	2	3	1	2

	1	2	3	4	5	6	7	8
A	~	~	0	~	0	~	~	~
B	1	3	2	~	1	3	2	3
C	3	1	2	2	3	2	2	1
D	~	1	3	3	~	1	3	2

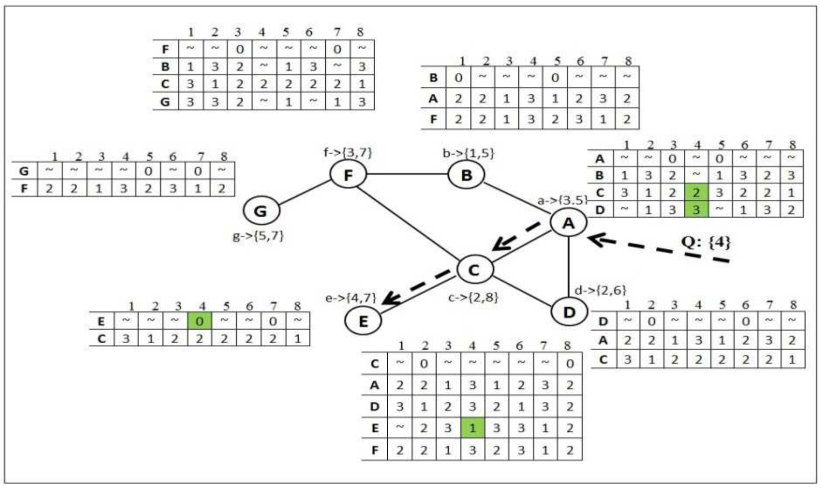
	1	2	3	4	5	6	7	8
E	~	~	~	0	~	~	0	~
C	3	1	2	2	2	2	2	1

	1	2	3	4	5	6	7	8
D	~	0	~	~	~	0	~	~
A	2	2	1	3	1	2	3	2
C	3	1	2	2	2	2	2	1

	1	2	3	4	5	6	7	8
C	~	0	~	~	~	~	~	0
A	2	2	1	3	1	2	3	2
D	3	1	2	3	2	1	3	2
E	~	2	3	1	3	3	1	2
F	2	2	1	3	2	3	1	2

Figure 2. Query routing in RDV by graph

The radius was set to 3, so nodes were only aware of the resources within 3 hops. In these tables, local resources were shown with 0. The symbol ~ showed that the length of path was more than the radius. In this method, when a request arrived at a node, first, its local resources were checked; if no appropriate resources for the request were found, its neighbor's resources from the RDV table were checked. In this example, node A received a request for resources 4. It checked its routing table and found two matches: through C with 2 hops (C4= 2) and through D with 3 hops (D4= 3). Since the shortest distance to the resource was 2 through neighbor C, the request was forwarded to C. Similarly, C forwarded the query to E. The node E found a match in its local vector and then its resource were assigned to the user that was the owner of request. Now, assuming that the request took the desired path, the node E was turned off. If the time to live (TTL) was not expired yet, it can be routed again. Thus, the node E was turned off and the routing tables of nodes must be updated, which was a very time consuming work on the graph. As shown in Figure 2, first, node C must update its own table and then send this information to its neighbors until updating its own tables, which is done again and again in other nodes. So, all nodes on the graph had to update their routing tables. On the other hand, as shown in Figure 2, the node C had 4 neighbors; so its routing table had 5 rows. When the grid network was extended and consisted of many nodes, considering that each node of this graph can have many neighbors, its routing table had many rows. Therefore, the search for resources of user's request took a lot of time and led to the expiration of TTL and increase of failed requests. Also, increasing the size of the table causes extreme memory consumption.



	1	2	3	4	5	6	7	8
F	~	~	0	~	~	~	0	~
B	1	3	2	~	1	3	~	3
C	3	1	2	2	2	2	2	1
G	3	3	2	~	1	~	1	3

	1	2	3	4	5	6	7	8
B	0	~	~	~	0	~	~	~
A	2	2	1	3	1	2	3	2
F	2	2	1	3	2	3	1	2

	1	2	3	4	5	6	7	8
G	~	~	~	~	0	~	0	~
F	2	2	1	3	2	3	1	2

	1	2	3	4	5	6	7	8
A	~	~	0	~	0	~	~	~
B	1	3	2	~	1	3	2	3
C	3	1	2	2	3	2	2	1
D	~	1	3	3	~	1	3	2

	1	2	3	4	5	6	7	8
E	~	~	~	0	~	~	0	~
C	3	1	2	2	2	2	2	1

	1	2	3	4	5	6	7	8
D	~	0	~	~	~	0	~	~
A	2	2	1	3	1	2	3	2
C	3	1	2	2	2	2	2	1

	1	2	3	4	5	6	7	8
C	~	0	~	~	~	~	~	0
A	2	2	1	3	1	2	3	2
D	3	1	2	3	2	1	3	2
E	~	2	3	1	3	3	1	2
F	2	2	1	3	2	3	1	2

Figure 3. RDVBT query routing

As seen in Figure 3, node Z received a request, found a path with 2 hops in its RDV table and forwarded the request through that to assign the node's resources to the request. Now, if the same assumption is considered, it can be observed that, in the worst case in a binary tree, the nodes only update to the root node. As can be seen in the previous method, when there is a need for updating the state of the nodes, each node and its neighbors should be checked because they are offline. Therefore, if there are N nodes, the number of updates is more than N times. While, using the RDVBT technique, the nodes are only checked up to the root node. Therefore, when there are N nodes in the network, the number of updates in the worst case will be equal to the logarithm of N. Also, the RDVBT method avoids as much as possible sending requests to unnecessary paths because, due to RDV tables, there is information about the nodes which have the resources required for the request.

```
1. Create tree
2. Initial rdv tables for each node
3. for each request do
4.      If the node has the requested resources in its local resources
5.          If the node is on then success; else failure and update the rdv tables
6.      Else
7.          If due to node's rdv table, the requested resources are not available in the node's subchilds
8.              If the node is the root then failure; else go to node's father and goto step 4
9.          Else
10.             Goto the node which has the requested resources with the minimum route
11.             If the resources are on
12.                 If TTL satisfied then success; else failure
13.             Else
14.                 Failure and update the rdv tables
15. End For
```

Figure 4. Algorithm for finding resources for any request

In this method, detecting the offline resources and updating was done when the request arrived at them. In this case, the offline resource left the network and the routing tables of nodes were updated. In fact, updating in this way was on demand. In other words, in such networks in which the average number of hops for reaching the resources or number of offline nodes is high; if the offline resources were not detected over time and their number increased, the number of visiting offline resources increased and the time of routing and average number of hops were enhanced as well. So, periodic detection of offline resources was used instead of the on demand one. Because when updating method was used in time distances, the offline resources were omitted from the available list resources and fewer requests faced with offline resources. Thus, the routing time and average number of hops to reach the resources decreased. But, this method had a serious problem because it imposed high traffic on the network during the updating time. Therefore, to avoid this, the number of offline resources was checked and, if it was more than ten percent of the nodes, the updating action would be done. So, the network did not incur high traffic and also the offline resources were omitted from the routing tables during the updating action.

It is used two methods for updating the routing tables: one of them is on-demand updating and the other one is periodic updating. In on-demand method, when the request arrived at node and was distinguishing that the resource is offline, the routing table will be updated. But in periodic one, the routing table of nodes is updating in specific time slices. In this paper, the combination of these two methods is used.

5. SIMULATION AND EXPERIMENTAL RESULTS

In order to evaluate the efficiency of the proposed method, MATLAB was used for simulation and its comparison with the resource distance vector graph mechanism. Proposed algorithm was executed by the personal computer with dual core processor and 4 GB RAM. First, the available resources were distributed uniformly and randomly among the nodes and also the users' requests were randomly distributed at different levels of the tree. To evaluate the performance, the experiment was done in two cases. In the first case, the network size was fixed and the experiment was repeated for the number of 100 to 1000 requests. In this part, the network size was 4000 in terms of nodes and the graph of each cluster consisted of 100 nodes. The number of successful requests and the number of requests that could not reach their needed resources were compared. Furthermore, the response time and the number of hops that were being spent in both methods were compared. As can be seen in Figure 5, the experiment was done for different numbers of requests and it was observed that the number of requests that were able to successfully find the resources in RDVBT was better than the ones in the previous method (RDV by graph).). Also, there are many similar paths in a graph, so when the request is checking these similar paths, it loses its TTL (time to live) and this leads to failure. However, because the unnecessary paths are removed in a tree, therefore more requests can find their required resources and the cost of failure is less than the previous method.

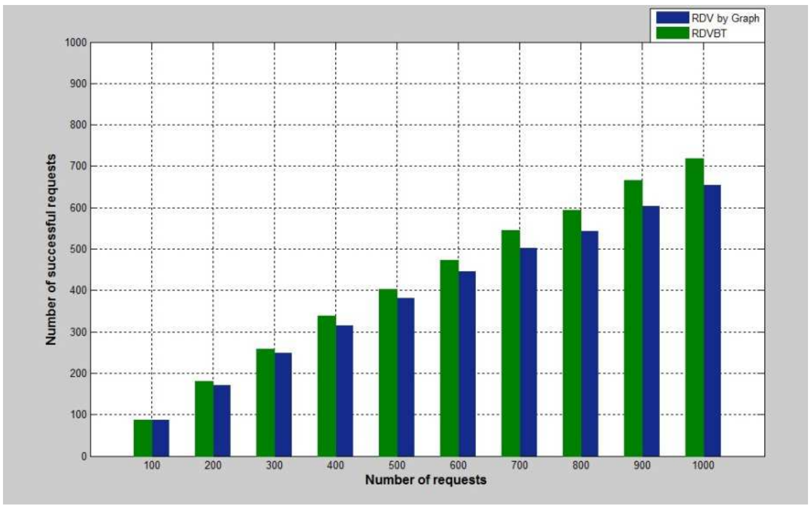


Figure 5. The number of request that being successful in reaching the resources

Figure 6 demonstrates the response time of both methods. It is observed that the time spent in RDV by the graph was much more than the time spent in RDVBT. In RDVBT, with increase in the number of requests, not much change can be observed in response time; but, in RDV by the graph, with the increase in the number of requests, the response time is often enhanced.

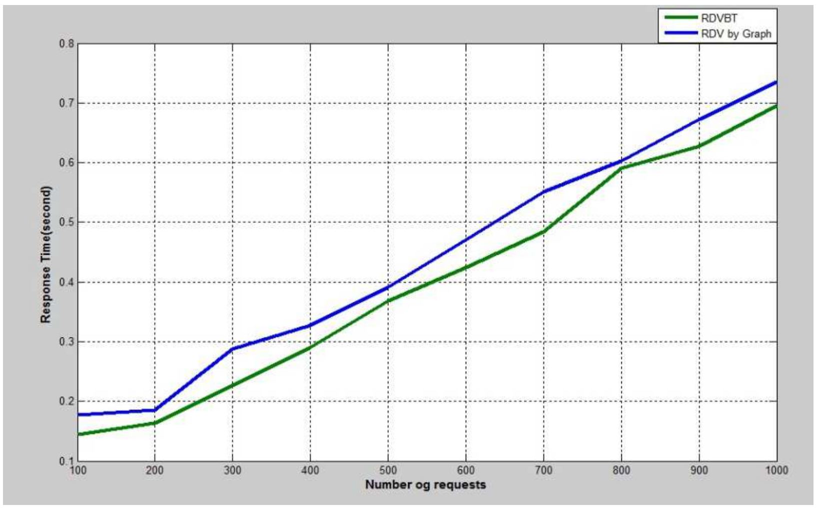


Figure 6. The response time of methods

Figure 7 also shows the number of hops that were spent to find resources for the requests in different numbers of requests. As can be observed, in RDV by the graph method, the requests go to unnecessary paths for finding resources; therefore, they consume time and make more failures than the RDVBT method.

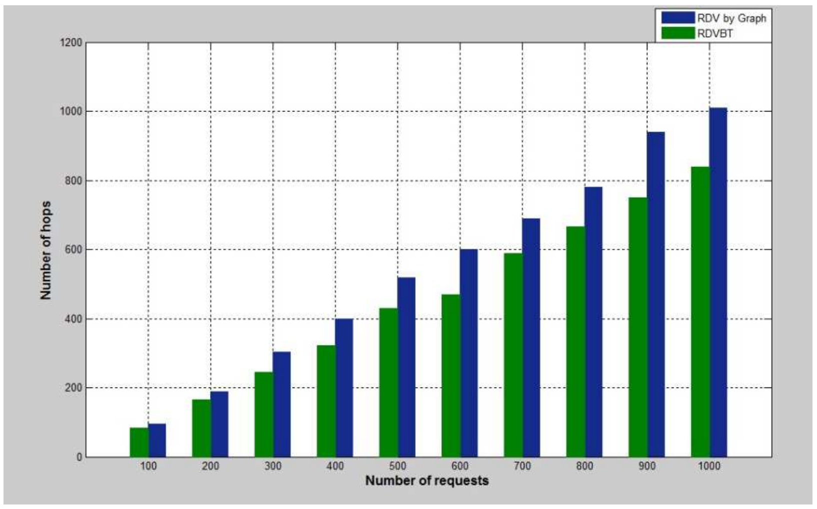


Figure 7. Number of hops that spent in both methods

In the second case, the number of requests was fixed at 1000 and the network size was increased from 4000 to 20000 in 5 steps. Again, the number of successful requests and the response time were compared in both methods and it was observed that the results of the proposed method (RDVBT) were better than those of the previous one. The results are shown in Figsure 8 and 9. Again, the number of hops was compared in both ways and it was depicted that, unnecessary paths were omitted in the RDVBT method rather than the RDV by graph one; therefore, finding the resources took less number of hops. Figure 10 shows the result of this part.

The experiments are done in three type of update for the different number of nodes at network. As it can be seen in table 1, if the response time of requests is important, on-demand updating method can be used. But, due to the less update and more faced with offline resources, the number of successful request is fewer than the other methods. On the other hand, if the number of requests which were found their required resources (successful requests) is important, periodic updating method is used.

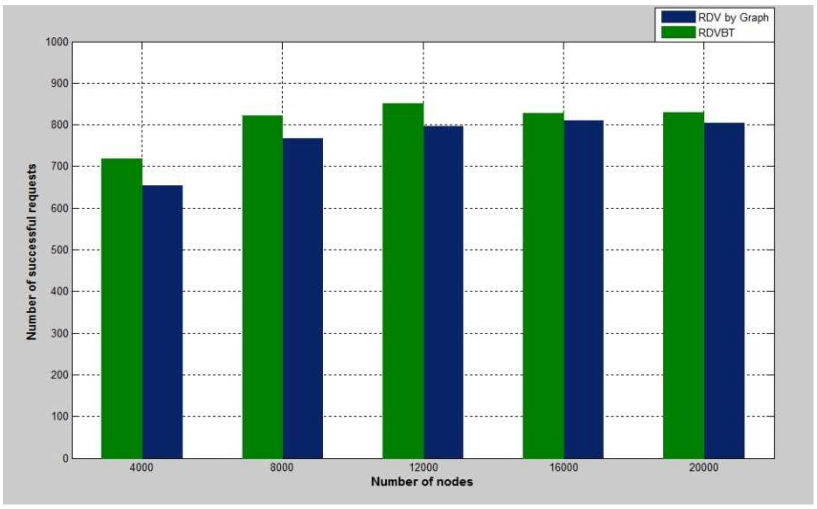


Figure 8. The number of request that being successful in reaching the resources

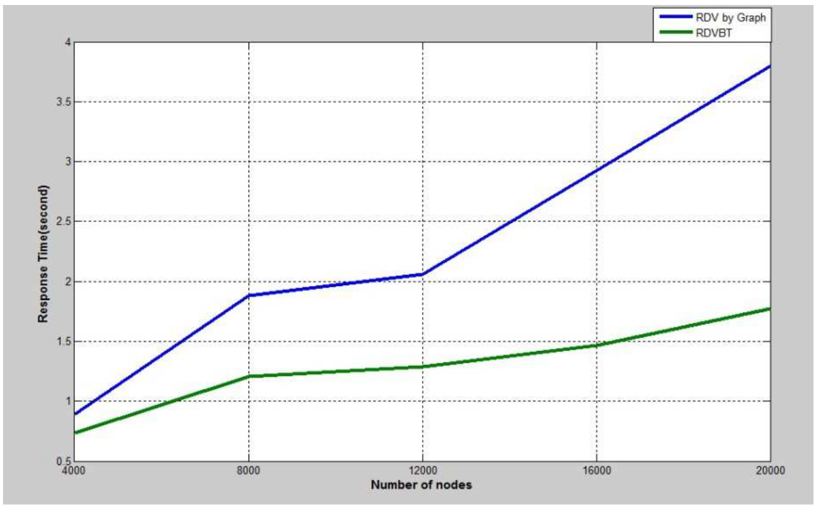


Figure 9. The response time of methods

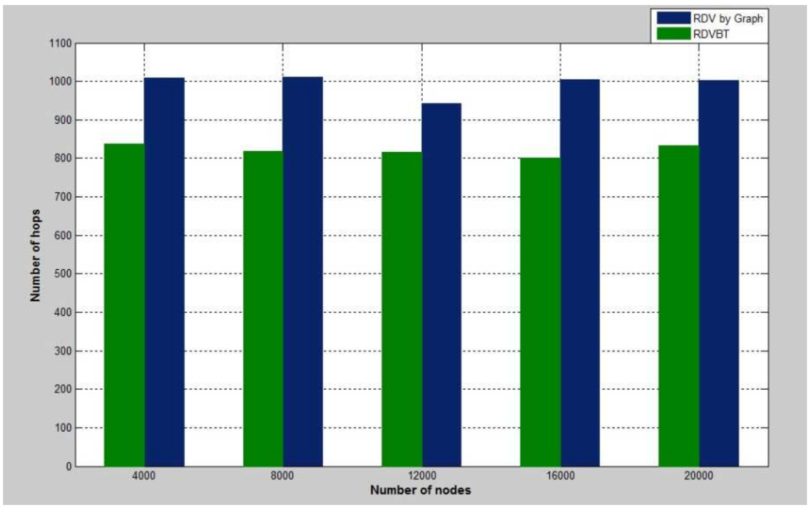


Figure 10. Number of hops that spent in both methods

When both of updating methods was used with together, the number of successful requests is more than each of them. As can be observed in table 1, in RDVBT method than the RDV by graph, the response time and also updating time is too low. Therefore, in RDVBT method, the requests can find their needed resources in low time and the type of updating is not effect on it.

Table 1. compare of RDVBT and RDV methods in terms of updating

Number of nodes in Grid network	**Type of update**	**Number of hops**		**Number of successful requests**		**Response Time**		**Updating Time**	
		Number of requests : 1000							
		RDVBT	**RDV**	**RDVBT**	**RDV**	**RDVBT**	**RDV**	**RDVBT**	**RDV**
4000	On-demand & periodic	**838**	1009	**719**	654	**1.174**	89.116	**0.438**	88.227
	On-demand	**868**	1014	**595**	521	**0.615**	39.943	**0.064**	39.060
	periodic	**810**	1026	**688**	650	**1.213**	55.530	**0.418**	54.714
8000	On-demand & periodic	**818**	1012	**822**	768	**2.247**	281.191	**0.747**	279.708
	On-demand	**859**	1036	**726**	679	**0.609**	90.261	**0.060**	89.036
	periodic	**822**	1015	**791**	742	**1.518**	214.456	**0.498**	213.229
12000	On-demand & periodic	**815**	942	**850**	797	**1.895**	591.433	**0.610**	589.373
	On-demand	**813**	956	**770**	722	**0.579**	151.177	**0.043**	149.522
	periodic	**795**	993	**839**	784	**1.817**	433.514	**0.564**	431.880
16000	On-demand & periodic	**800**	1005	**827**	810	**2.073**	927.590	**0.608**	924.666
	On-demand	**824**	1021	**762**	741	**0.589**	157.259	**0.031**	154.914
	periodic	**820**	1018	**820**	793	**2.171**	762.643	**0.622**	760.262
20000	On-demand & periodic	**833**	1002	**829**	804	**2.429**	1264.2	**0.659**	1260.4
	On-demand	**846**	1058	**765**	750	**0.676**	139.469	**0.029**	135.326
	periodic	**802**	993	**818**	792	**2.313**	1108.7	**0.629**	1105.8

6. CONCLUSION

In this paper, a binary tree was applied instead of the graph in RDV method to decrease the time of finding resources and time of updating routing tables. This method made it possible to avoid going to extra and unnecessary nodes and also decreased the time of updating routing tables and response. Simulation results showed the number of successful and failed requests and the number of hops that were visited in resource discovery, which demonstrated that response and updating time of the tree (RDVBT) was less than those of the previous method (RDV by graph). In future works, there should be a focus on the optimization of the systems and this can be followed by adding parameters and implementing the algorithm on a splay tree in order to decrease the number of hops and get better results.

REFERENCES

[1] Yang K., Guo X., Galis A., Yang B., & Liu D., "Towards efficient resource on demand in Grid computing". *Operating Systems Review*, (2003), 37(2), 37–43.

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

[3] Juan Li, Son Vuong, "A Scalable Semantic Routing Architecture for Grid Resource Discovery", Proceedings of the *2005 11th International Conference on Parallel and Distributed Systems*, (ICPADS'05)

[4] Juan Li," Grid resource discovery based on semantically linked virtual organizations", *Future Gener. Comput. Syst.* 26 (2010) 361_373

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

[6] M. Mutka, M. Livny, "Scheduling remote processing capacity in a workstation processing bank computing system", in: *Proc. of ICDCS, September* 1987.

[7] C Germain, V Neri, G Fedak, F Cappello, XtremWeb: "Building an experimental platform for global computing", in: *Proc. of IEEE/ACM Grid*, December 2000.

[8] A Chien, B Calder, S Elbert, K Bhatia, Entropia," Architecture and performance of an enterprise desktop grid system", J. *Parallel Distrib. Comput*. 63 (5) (2003).

[9] F Berman, et al., “Adaptive computing on the grid using AppLeS”, *TPDS* 14 (4) (2003).
[10] 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.
[11] S. Tangpongprasit, T. Katagiri, H. Honda, T. Yuba, “A time-to-live based reservation algorithm on fully decentralized resource discovery in grid computing”, *Parallel Comput*. 31 (6) (2005) 529–543.
[12] R.-S. Chang, M.-S. Hu, “A resource discovery tree using bitmap for grids”. *Future gener. Comput. Syst*. 26 (2010) 29–37.

BIOGRAPHIES OF AUTHORS

Seyed Elyar Hashemsereshtreceived his B.S. degree in ComputerEngineering from Azad university of Shabestar, Iran, in 2007.He has been a M.S. student at the Azad university of Tabriz, Tabriz, Iran, since 2009. His research interests include gridcomputing and wireless sensor networks.

Ali Asghar Pourhaji Kazem received a B.Sc. degree in computerengineering from University of Isfahn and also a M.S. degree in computer engineeringfrom Shahid Beheshti Universityin Tehran. He is currently a Ph.D. student of computer engineering in Science and Research branch of Islamic Azad Universityin Tehran. Hiscurrent research interests include distributed systems, Grid computingand Cloud computing.