International Journal of Computational Intelligence and Informatics, Vol. 2: No. 4, January - March 2013

Efficient Memory Utilization for Resource Discovery in a Grid Environment

D. Ramyachitra
Assistant Professor,
Department of Computer Science,
Bharathiar University,
Coimbatore - 641046
Tamil Nadu, India.
jaichitra1@yahoo.co.in

S A SharmilaBanu
Research Scholar
Department of Computer Science,
Bharathiar University
Coimbatore
Tamil Nadu, India
shahabanu18@yahoo.com

***Abstract*- One of the fundamental requirements of grid computing is efficient and effective resource discovery mechanism. Resource discovery involves discovery of appropriate resources required by user applications. In this regard various resource discovery mechanisms have been proposed during the recent years. Simple matchmaking rules are used to identify each resource as a part of a certain technical category and the distance travelled in hops is calculated for a certain request. This paper deals with the reduction in the time taken for matching the resources based on the user's requirements. In this paper Matchmaking algorithm, Flooding algorithm, Swamping algorithm, Random pointer jump algorithm were applied for hops calculation and for efficient memory utilization of the routers. From the simulation results it is found that swamping algorithm gives better result when compared to Flooding, Random pointer jump and matchmaking algorithms. To lessen the burden on the routers, partial information of virtual organization is stored in the routers. This results in efficient memory utilization on the routers. Comparative graphs are also given to show the efficient utilization of memory space and hops calculation of the resource of the routers.**

Keywords: Resource Discovery, Virtual Organization, Reliability, Flooding algorithm, Swamping algorithm, Random pointer jump algorithm.

I. INTRODUCTION

Grid computing provides an effective infrastructure for massive computation among flexible and dynamic collection of divided system for resource discovery. Grid system is a large scale of distributed environment which provides a high number of powerful resources to its users. It provides many types of resources such as computing resources (CPU cycles, storage, network bandwidth and memory etc) and services (access to specific data, shared software etc) [1]. Resource Discovery is a systematic process of determining which grid resource is the best candidate to complete a job with trade-offs like i) shortest amount of time ii) most efficient use of resources iii) minimum cost. Resource discovery in a grid system can be defined as searching and locating resource candidates which are suitable for executing jobs in a reasonable time in spite of the dynamicity and large scale of the environment [2]. Success of grid system mainly relies on efficient usage of the right resources. Resource discovery is an important step in finding these resources. A number of derivatives of grid computing exist such as data grid, compute grid and science grid etc [3]. In recent years there has been a large increase in grid technologies research, which has produced some reference to grid implementations. A vast number of researchers have been putting in a lot of effort to facilitate building and efficient utilization of grids [4].

Fig 1 shows the components and methods of grid resource discovery which includes resource selection, scheduling, grid market directory, data, data node, grid, grid info, NI grid, globus, Alchemi etc. Resource Discovery schema maintains and queries a resource database known as "status" database. This database is maintained network wide to fulfill the client request information service. Discovery is initiated by a network application to find suitable resources within the Grid. For this it has to interact with the resources individually through agents or it maintains the information about the resources in databases.

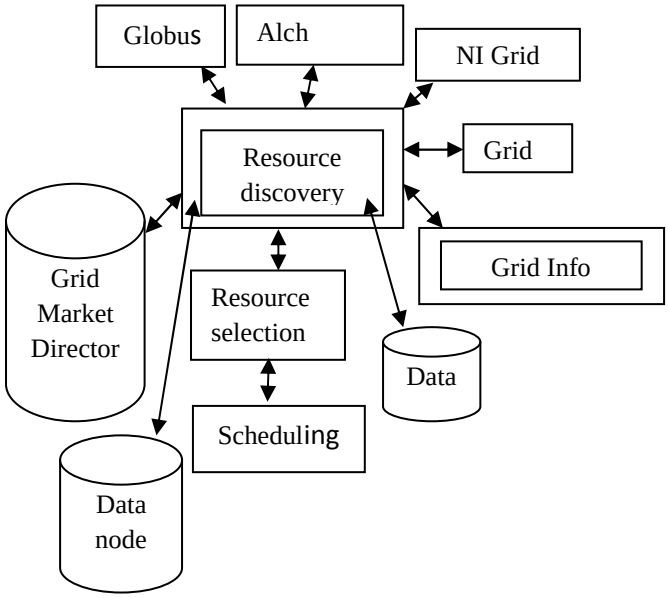

Figure 1: Grid Resource Discovery [5]

The grid resource discovery problem [6] can be defined as the problem of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements. The problem is complicated by the fact that some resource information (e.g., CPU load or available storage) changes dynamically. Resource discovery techniques maintain the resource attribute and status information in a distributed database and differ in the way they update, organize, or maintain the distributed database. The challenge is to devise highly distributed discovery techniques that are fault tolerant and highly scalable. Matching the needs of an application with available resources is one of the basic and key aspects of a Grid system [7].

II. RELATED WORK

In this section, research work concerning the grid resource discovery problem in general, as well as implementation of trust mechanisms is presented. Though significant work has been done towards the direction of trusted grid computing [7] focusing to the security implications that could arise in such systems, resource discovery in grid system is limited. In this section related work regarding resource discovery implementations are presented.

The amount of communication required by the algorithm is measured by: (a) the *pointer communication complexity* defined as the number of pointers exchanged during the course of the algorithm, and (b) the *connection communication complexity* defined by the total number of connections between pairs of entities.
The *flooding* algorithm assumes that each node *v* only communicates over edges connecting it with a set of initial neighbors. In every round node *v* contacts all its initial neighbors and transmits to them updates [8]. And then updates its own set of neighbors by merging with the set. The number of rounds required by the flooding algorithm is equal with the diameter of the graph. The main advantage of flooding is the increased reliability provided by this routing method. Since the message will be sent at least once to every host it is almost guaranteed to reach its destination. In addition, the message will reach the host through the shortest possible path. Disadvantage of this algorithm is it can be very slow if not started with a graph, which has small diameter.

The S*wamping* algorithm allows a machine to open connections with all their current neighbors and not only with the set of initial neighbors. The graph of the network known to one machine converges to a complete graph on steps but the communication complexity increases. Here *n* is the number of nodes in the network [8]. The main advantage of this algorithm is this algorithm needs O (log n) rounds to converge to a complete graph and which is irrespective to the initial configuration. The disadvantage is communication complexity of this algorithm grows very quickly.

In the *random pointer jump* algorithm each node *connects* a random neighbor; neighbor's nodes will in turn merge [8]. A version of the algorithm called *the random pointer jump with back edge* requires nodes to send back to a pointer to all its neighbors. There are even strongly connected graphs that require with high probability time to converge to a complete graph in the random pointer jump algorithm.

Matchmaking (also called Matchmaker) [9] is a distributed resource management mechanism developed as part of Condor [10] project for Grid systems. In a simple view, the matchmaking acts as"yellow page" service to enable tasks to find resources for execution. The matchmaking can support dynamic clusters, i.e. resources can enter and join their local clusters at will. The matchmaking is based on the idea that resources providing services and clients requesting service advertise their characteristics and requirements using classified advertisements (*classads*). The classad specification defines the syntax and semantic rules for specifying the attributes associated with the characteristics and requirements. It may be possible to use the classad language as the specification language as the part of resource discovery mechanism in Grid systems. In [12], O. F. Rana, et.al., proposed a system that used the matchmaking in Computational Grid.

A noteworthy approach to the resource discovery problem is the matchmaking framework [12]. The matchmaking framework was designed to solve real problems encountered in the deployment of condor, a high throughput computing system. Several other research papers make use of the matchmaking framework trying to add new aspects in the existing mechanisms. According to this framework, requestors and providers (resource) in a grid system advertise their characteristics. A matchmaking service is responsible for finding a match between the advertisements and informing the relevant entities of the match.

Another notable approach to the resource discovery problem is the semantic community's one [13]. Motivation behind the semantic communities approach is that grid communities and human consists of members that are engaged in sharing and communication. Main target in this approach is to create grid communities based on similar- interest's policies allowing community nodes to learn of each other without relying on a central meeting point.

In the eight trusts model [14] the global reputation of each peer is given by the local reputation values assigned to this peers by other peers, weighted by the global reputations of the assigning peers. Normalizing local reputation values in a sensible manner so that malicious peers cannot easily subvert the system and using an efficient algorithm to aggregate the local reputation values, the model concludes to the global reputation value of the peer.

In [15], an ontology based Matchmaker Service is proposed that supports dynamic resource discovery and resource descriptions. However, the request is expressed using request ontology and hence there is a need to compile the user request as ontology descriptions. In [16] both resources and the content stored at peers are described by means of RDF metadata. Routing indices located at super-peers use such metadata to perform the routing of queries expressed through the RDF-QEL query language. Puppin et al. [17] proposed a Grid Information Service based on the super-peer model and its integration within OGSA. The Hop Counting Routing Index algorithm is used to exchange queries among the super-peers and in particular to select the neighbor super-peers that offer the highest probability of success.

The Matchmaking framework [18] was designed to solve real problems encountered in the deployment of Condor, a high throughput computing system. Several other research papers make use of the Matchmaking framework trying to add new aspects in the existing mechanism [19]. According to this framework, requestors and providers (resources) in a Grid system advertise their characteristics. A matchmaking service is responsible of finding a match between the advertisements and informing the relevant entities of the match.
The P2P Grid is emerging as a promising platform for executing large-scale, resource intensive applications. Iamnitchi et al. proposed resource discovery approach in [20] based on an unstructured network similar to Gnutella combined with more sophisticated query forwarding strategies taken from the Freenet overlay network. Several systems exploiting DHT-based P2P approaches for resource discovery in Grids have recently been proposed [21].

III. PROPOSED WORK

In this paper resource discovery algorithms are used for hops calculation. In existing system matching the request with the neighbor routers are a little overhead. The Matchmaking, Flooding, Swamping and Random Pointer Jump algorithms are used for matching the resources with the user's request. An attempt has been made to reduce the time taken for matching the resources and also for efficient utilization of memory on the routers.

Creation of Virtual Organization

Virtual organizations (VO) are created based on different categories that depend upon disk size, memory size and processor speed. Table 1 shows the summary of categories used in the simulation.

TABLE 1. SUMMARY OF CATEGORIES

Categories	Disk Size(MB)	Memory Size(MB)	Processor Speed(GHz)
Very Limited function	<50000	<128	<0.5
Limited function	50000 to 100000	256 to 512	1 to 1.5
Typical	100000 to 150000	512 to 1024	1.5 to 2
Competent	150000 to 200000	1024 to 2048	2 to 2.5
Very Competent	>200000	>2048	>2.5

Grades are assigned based on the size of the disk and memory and the speed of the processor. The disk that has the smaller size is assigned as grade 1 and that has the larger size is assigned as grade 5. Likewise for memory with smaller size is assigned as grade 1 and of larger size is assigned as grade 5. The same procedure is applicable for processor also. i.e., the processor with speed in low assigned as grade 1, and the high speed is assigned as grade 5. The technical category eqn (1) is calculated using disk grade and memory grade. In the proposed work, as processor also plays an important role in the computation, processor grade is also taken into account for the calculation of technical category and it is shown in the eqn(1).

Technical category= (round (disk grade + memory grade+processor grade)/2) (1)

Reliability value is assigned to each system separately. In the beginning, reliability value is assigned as 0.1 for all the system equally. In the later stages, this value might increases or decrease depending on the execution of user tasks successfully. Using this reliability, the virtual organization can be either identified as complete trustworthy or partial trustworthy. If reliability of each system is greater than 0.75, then the trustworthy value of that virtual organization is the average of the reliability value of all the systems. Trustworthy of a virtual organization is assigned in the range 0 to 1. Based on the user's request, the algorithms explained in the following sections match the resources with the user's request based on the trustworthy of the virtual organization and their technical category.

Matchmaking algorithm

Matchmaking algorithm is used in grid environment to discover resources. Let us consider there are 6 virtual organization and all these virtual organizations are connected through routers. Each router holds the respective virtual organization information. Randomly global trust value is allocated to each virtual organization. User sends the request to the virtual organization for resource allocation.

Algorithm for Matchmaking

Input: Virtual_id, Request type

Output: Updated value

1. Each virtual organization connected to the router.
2. Generation of resource request by VO
3. Forward the request to corresponding VO
4. Updation of global trust values of VO on satisfaction of the request
5. Display of hops value for the request

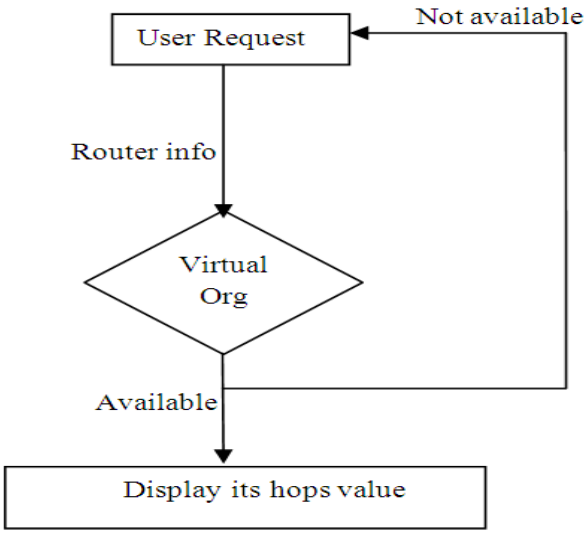

Figure 3. Flow chart for Match making Algorithm

Matchmaking algorithm is used to satisfy the user request. Using this algorithm VO check its own router information such as resource type, if it matches then it will display the hops value and update the global trust value. If not, it sends the acknowledgement to the user as request failed. It checks the request only with the connected router but not with the neighbor router. This leads to a disadvantage for matchmaking algorithm.

Flooding Algorithm

This algorithm is widely used by internet routers and every node acts as a transmitter and receiver, every node tries to send every message to every node of its neighbors except the source node. In the flooding algorithm, a node is initially configured to connect with a fixed set of neighboring nodes. It can only communicate directly to this set. We concern the nodes in the network connected in the form of graph structure.

A node only communicates over the edges that were initially in the graph (or initial neighboring nodes); new edges that are added to the graph are not used for communication. In the Flooding algorithm in every round, each node contacts all of its initial neighbors (neighboring nodes) and transmits the updates to them. The example of update information is new nodes that are joined before sending the updates.

The hops value is calculated for the user's request using the global trust values. System architecture for the above generalized algorithm of the flooding method for resource discovery is shown in fig 5.

FLOODING ALGORITHM FOR HOPS CALCULATION

INPUT: Virtual_ id, No of Requests, request type

OUTPUT: Hops value

1. User's request is given as input to the router
2. Router compares the user's request with its virtual organization
3. If found, it displays its hops value
4. Otherwise Router checks the request with neighbor's virtual organization and displays its hop's value.
5. Repeat the steps 1 to 4 until all the requests are completed

Figure 4. Flooding algorithm for hops calculation

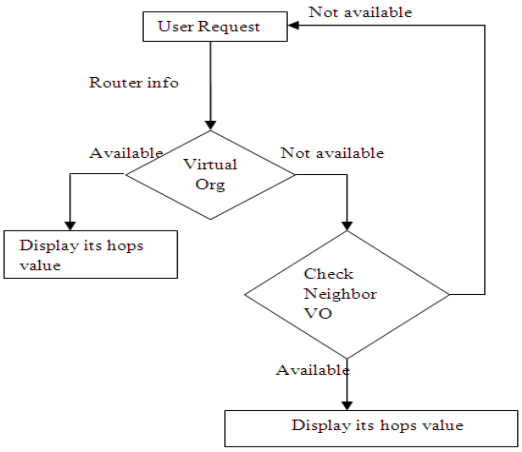

Figure 5 Flow chart for Flooding algorithm

Swamping Algorithm

Swamping algorithm is similar to flooding algorithm except, this algorithm allows a node to connect with all of its current neighbors, not only with the set of initial neighbors. The speed of the swamping algorithm is better than the flooding algorithm when it determines their ability of converging to a complete graph.

A machine sends requests to every machine which are its neighbors currently and initially. Each of n machines makes connections with each of its n neighbors. Here communication complexity grows quickly. In this work, the request from virtual organization is compared with all of its neighbors, if not found it is send to the neighbors of neighbors.

Using this swamping algorithm, the better hops value for the request is calculated. System architecture for the above generalized algorithm of the swamping method for resource discovery is shown in fig 7.

SWAMPING ALGORITHM FOR HOPS CALCULATION

INPUT: Virtual_ id, No of Requests, Request type

OUTPUT: Hops value

1. User's request is given as input to the router
2. Router compares the user's request with all of its virtual organization
3. If found, it displays its hops value
4. Otherwise Router checks the request with neighbor's of neighbors' virtual organization and displays its hop's value.
5. Repeat the steps 1 to 4 until all the requests are completed

Figure 6. Swamping Algorithm for hops calculation

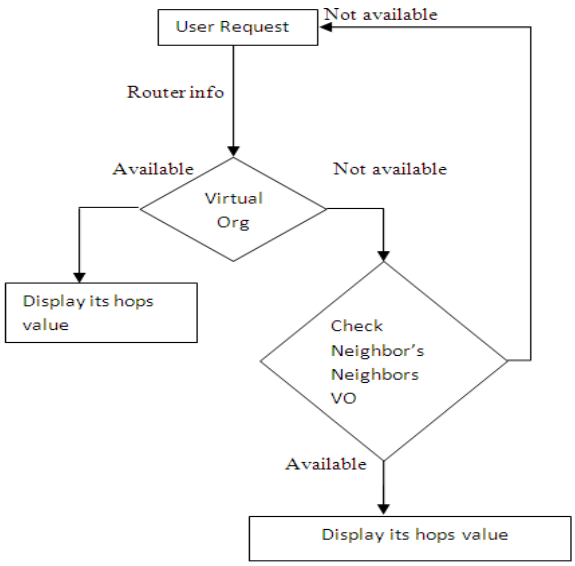

Figure 7. Flow chart for Swamping Algorithm

Random Pointer Jump Algorithm

In this algorithm in each round, each node contacts with a random neighbor, and then this random neighbor sends all of its neighbors to the sender node. Finally sender neighbor and random neighbor's neighbors get merged. The random pointer jump algorithm can be only applied to strongly connected networks i.e. there must exist a path between every pair of machines, otherwise the graph will never converge to a complete graph.

Using this random pointer jumping algorithm, the better hops value for the request is calculated. System architecture for the above generalized algorithm of the random pointer jumping algorithm method for resource discovery is shown in Fig 9.

RANDOM POINTER JUMP ALGORITHM FOR HOPS CALCULATION

INPUT: Virtual_ id, No of Requests, Request type

OUTPUT: Hops value

1. User's request is given as input to the router
2. Router compares the user's request randomly with one of its neighbor's virtual organization
3. If found, it displays its hops value
4. Otherwise Router checks the request with neighbor's of neighbors' virtual organization and so on
5. Repeat the steps 1 to 4 until all the requests are completed

Figure 8. Random pointer jump Algorithm for hops calculation

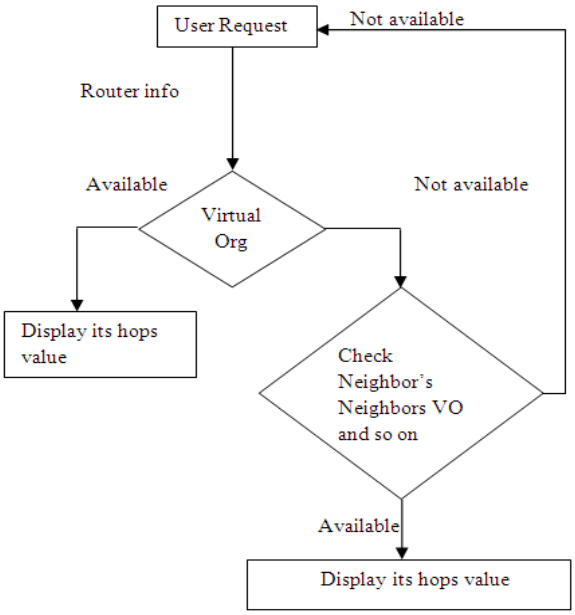

Figure 9. Flow chart for Random pointer jumping algorithm

IV. SIMULATION RESULTS AND DISCUSSION

The simulation result shows that the proposed method of memory utilization is more efficient than the existing method. Comparative graphs are also given to show the efficiency of the Memory utilization and distance calculation of the request. The comparative analysis based on memory capacity and distance calculation is shown through the tables 2 and 3, figure 10 and 11.

Performance comparison of matchmaking, flooding, swamping and random pointer algorithm for finding hops

The algorithms such as Matchmaking, Flooding, Swamping and Random pointer jump algorithm performs the hops distance calculation for the resource. It is found that hops value for the swamping algorithm is less compared to all the algorithms.

From the simulation results it is found that,

- The flooding Algorithm reduces the hops distance from 6 % to 25% compared to the matchmaking one.
- The Swamping Algorithm reduces the hops distance from 2 % to 50% compared to the matchmaking one.
- The Random pointer jump Algorithm reduces the hops distance from 7 % to 40% compared to the matchmaking one.
- It is found that the swamping algorithm performs better for most of the user's request. When swamping algorithm is compared with matchmaking algorithm, there is a fluctuation in the performance i.e. for 5 request, there is a 50% reduction in hops value for swamping algorithm, for 10 requests approximately 10% reduction, for 20 requests 20% reduction, for 30 requests 10% reduction, for 40 requests 20% reduction, for 50 requests 3% reduction. This fluctuation is applicable for the comparative performance of the swamping algorithm for the remaining algorithms also.

- Even though there is a fluctuation in the performance, it is found that Swamping algorithm performs better compared to all the other proposed algorithm

TABLE 2: PERFORMANCE COMPARISON OF MATCHMAKING ALGORITHM, FLOODING ALGORITHM, RANDOM POINTER JUMP ALGORITHM AND SWAMPING ALGORITHM AND FOR FINDING HOPS.

NUMBER OF REQUESTS	HOPS			
	MATCHMAKING ALGORITHM	FLOODING ALGORITHM	RANDOM POINTER JUMP ALGORITHM	SWAMPING ALGORITHM
5	20	15	12	10
10	30	25	24	26
20	50	43	39	40
30	85	80	82	78
40	105	90	90	85
50	102	100	95	99

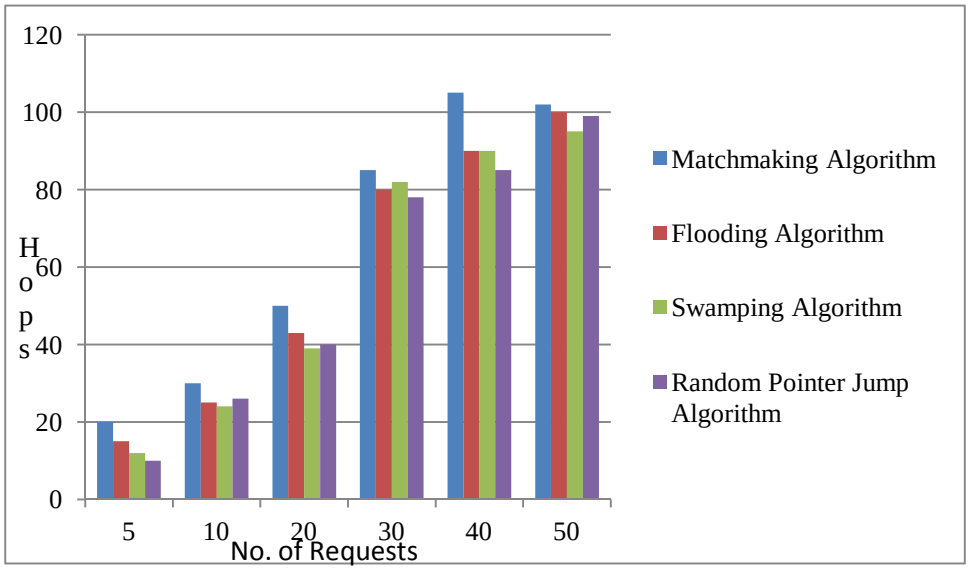

Figure 10. Performance comparison of Matchmaking algorithm, flooding algorithm, swamping algorithm and random pointer jump algorithm for finding hops

TABLE 3. PERFORMANCE COMPARISON OF MATCHMAKING ALGORITHM AND SWAMPING ALGORITHM FOR MEMORY USAGE

DISK SIZE (MB)	USAGE OF DISK CAPACITY (MB)	
	MATCHMAKING ALGORITHM	SWAMPING ALGORITHM
50000	42000	39000
100000	78000	68000
150000	115000	100000
200000	150000	120000
250000	225000	199000
300000	290000	250000

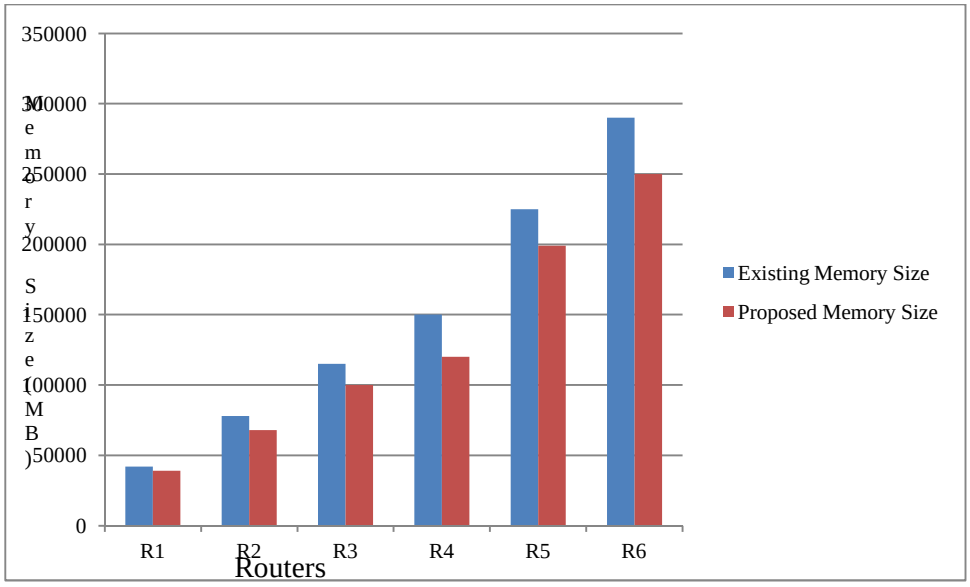

Figure 11. Performance comparison of matchmaking algorithm and swamping algorithm for usage of memory.

Performance Comparison of Original Memory size with Existing Memory size

In the proposed work there are six routers. Each router is connected to its virtual organization. Routers have only partial information about the corresponding virtual organization and not the whole. Here the memory space for each router is calculated and thus the values range from 50000 to 300000 MB.

Thus for the disk size from 50000 to 300000 MB, there is a decrease of 8% to 20% in the memory size of the swamping algorithm compared to the Matchmaking algorithm. Swamping algorithm provides the better results for memory utilization also. Thus it is concluded that Swamping algorithm performs well in terms of hops calculation as well as memory utilization.

V. CONCLUSION

In this paper, we have used resource discovery algorithms for the calculation of hops for the resource as well as for efficient memory utilization. From the simulation results, it is found that swamping algorithm performs better compared to the remaining algorithms. Also, the proposed method stores only partial information of virtual organization and hence memory space is reduced.

REFERENCES

[1] J. Joseph and C. Fellenstein, “Grid Computing”, IBM Press Pearson Education.
[2] Foster, I. and C. Kesselman,“The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers.
[3] K.Vivekanandan, D.Ramyachitra, “Bacteria Foraging optimization for protein sequence analysis on the grid”, Future generation computer systems, Vol 28, issue 24, April 2012.
[4] K.Vivekanandan, D.Ramyachitra, B.Anbu, “Artificial Bee Colony algorithm For Grid Scheduling”, Journal of Convergence Information Technology, vol 6, No 7, July 2011, pp.328-339.
[5] Iamnitchi and I. Foster, “On Fully Decentralized Resource Discovery in Grid Environments”, IEEE International Workshop on Grid Computing,Denver, CO, 2001.
[6] Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, “A taxonomy and survey of grid resource management systems for distributed computing”, John Wiley & Sons, Ltd. 17 September 2001.

[7] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems", School of Computer Science and Software Engineering,Monash University Melborne Australia.
[8] W. Li, Z. Xu, F. Dong, J. Zhang, "Grid Resource Discovery Based on a Routing-Transferring Model",Proc. of Third International Workshop on Grid Computing: GRID 2002, Baltimore, MD. , pp: 145-156, Springer, 2002
[9] Song.S. , Hwang, K, Kwok.Y, "Trusted grid computing with security binding and trust integration",Journal of Grid Computing 3(1), 2005.
[10] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor, "A hunter of idle workstations, 8th International Conference on Distributed Computing Systems", pp. 104-111, 1988.
[11] M. Maheswaran and K. Krauter,"A Parameter-based Approach to Resource Discovery",First IEEE/ACM International Workshop, 2000.
[12] O. F. Rana, D. Bunford-Jones and D. W. Walker,"Resource Discovery for Dynamic Clusters in Computational Grids",15th IEEE Internal Symposium on Parall and Distributed Processing, pages 759-767, 2001.
[13] Raman.R, "Matchmking frameworks for distributed resource management", Technical report, University of Wisconsin-maddison, 2001.
[14] Tangpongprasit,S., Katagiri, T., Honda, H., Yuba, T, "A time-to-live based reservation algorithm on fully decentralized resource discovery in grid computing", Parallel Computing 31, 2005,pp.529-543.
[15] Harth, Y. He, H. Tangmunarunkit, S. Decker, C. Kesselman, "A Semantic Matchmaker Service on the Grid", WWW2004, May 17-22, 2004, pp.326-327.
[16] Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A.: Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks, Proc. of World Wide Web Conference, Budapest, Hungary, 2003,pp.536-543
[17] Puppin D, Moncelli S, Baraglia R, Tonellotto N, Silvestri, "A Peer-to-peer Information Service for the Grid, Proc. of International Conference on Broadband Networks", San Jose, CA, USA (2004)
[18] R. Raman, M. Livny, M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing", hpdc, p. 140, Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7 '98), 1998.
[19] Rajesh Raman, "Matchmaking frameworks for distributed resource management", University of Wisconsin- Maddison, 2001.
[20] A. Iamnitchi and I.T.Foster, "On fully decentralized resource discovery in grid environments", proceedings of the Second International Workshop on Grid Computing, London, UK, Springer-verlag, 2001. pp.51-62.
[21] Andrzejak, A., Xu, Z., Scalable, "Efficient Range Queries for Grid Information Services", Proc. of 2nd IEEE Int. Conf. on Peer-to-peer Computing (P2P'02), Sweden, 2002.