# Distributed Resource Discovery in Wide Area Grid Environments

T.N. Ellahi and M.T. Kechadi

Parallel Computational Research Group,
Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland
{tariq.ellahi,tahar.kechadi}@ucd.ie

**Abstract.** Multitude of resources pooled in grid environments require a scalable, efficient, low-cost and self-organizing resource discovery mechanisms. P2P networks possess overlapping characteristics like large scale resource sharing and dynamic nature of participation. P2P style resource discovery approaches have been employed in the grid environments. We propose a P2P based resource discovery approach with modified features like customized neighbourhood defined according to the user preferences and distributed search using directed flow of requests contrary to the blind propagation of P2P systems.

## 1 Introduction

Substantial advancement in networking and hardware technologies have resulted in using computers for modelling and simulating complex scientific and engineering problems, diagnosing medical conditions, forecasting the weather and many other purposes. Yet, although there are certain challenging problems that exceed the existing technical ability to solve them. The technological advancements have led to the possibility of using wide-area distributed computing infrastructure for solving large-scale sophisticated problems.

Grid environments may consist of millions of resources pooled together by thousands of individuals and organizations shared under locally defined sharing policies to form Virtual Organizations [1]. Resources shared in virtual organizations include computers, storage elements, software applications, data sets, custom scientific instruments etc. Such multitude of resources shared in grid environments dictate the availability of a resource discovery mechanism, which can locate resources according to the user specified criteria. The Resource discovery is made challenging by the absence of a global resource information repository, resource heterogeneity and dynamic nature of participation. We are proposing a purely decentralized resource discovery solution, which equips the users with the capability to customise their environments according to their preferences and performance requirements.

The rest of the paper is structured as follows: Section 2 talk about the related work in this field. Section 3 and 4 discuss about the customization of neighborhood. Distributed search is presented in Section 5 and results in Section 6. Section 7 concludes the paper.

## 2 Related Work

Different systems have adopted different approaches for resource discovery. Globus uses MDS [2] which is LDAP based and hierarchical. GRIS provides information about resources and GIIS accumulate information from GRIS and GIIS of lower level. Users can query both components to obtain information about resources. Legion uses Collections [3] to store and lookup resource information.
Among P2P systems, Napster used a centralized directory of resource information. Gnutella used query flooding to discover resources. Both [4] [5] have suggested alternative request forwarding strategies. [6] uses routing indices to forward queries. Distributed Hash Tables (DHT) which include [7] [8] [9] [10] assign unique identifiers and organize nodes into a search-optimized overlay. Queries are forwarded based on identifier proximity.
[11] proposed P2P style resource discovery in grid environments with four alternative request forwarding strategies. [12] extends CAN but uses multiple indices for multiple attributes of the resources. [13] propose Chord based system with a single index.
Among approaches which have similarities with our approach, [14] adds shortcut on top of Gnutella topology based on locality of interest. Failure to find resource through shortcuts result in using underlying Gnutella topology. In [15] authors proposed the idea of using trader-based approach. Traders connect to other traders based on their preferences and forward the locally unsuccessful queries to other neighbouring traders.

## 3 Problem Statement and Model

Large scale resource sharing environments are characterized with the following properties:

– Existence of very large number of distributed resources shared by thousands of participants.
– Dynamic nature of participation and great deal of heterogeneity of resources and the sharing policies.
– Absence of a global naming scheme or a global repository hosting information about the resources. Every node is aware of a subset of nodes. This subset is called the neighbourhood of that node.
– Requests for the resources by specifying the desired attributes rather than the name of the resource.

We are proposing a P2P based resource discovery approach which besides coping with the above mentioned properties also provides to the user the ability to customise their neighborhood according to their preference. Following is the general overview of our approach.

*Every node specifies its preferences and performance requirements. Its neighbourhood will be selected among the nodes it knows about. When requests are* *issued, local neighbourhood will be searched first. As neighbourhood of every node is tailored to its needs, there is a high probability that resources will be discovered within the neighbourhood. If unsuccessful, request will be forwarded to a subset of the neighbourhood and those nodes can start the discovery in a recursive manner.*

The above approach works in a distributed and decentralized fashion and discovers the desired resources but raises two issues which needs to be resolved which are: selection criteria of the neighbourhood and selection of neighbours to forward the queries. Next two sections discuss both issues.

## 4 Neighbourhood

Contrary to the existing P2P systems in which the neighbourhood connectivity is either random or system imposed, our system adopts a different methodology for creating the neighbourhood connectivity. We believe that the participants of a P2P system in general and grid environments in particular have fixed preferences and performance requirements. So if the neighborhood of a node is selected based on user's preference, there is a high probability that user requests can be satisfied in a single lookup. We need to define some elements before we give a formal definition of neighborhood.

### 4.1 Distance Measure (Norm)

As stated earlier neighbourhood customisation is done by selecting nodes, which satisfy a user requirements. This leads to two questions, which are:

– what criteria should be used to measure the performance of a node?
– what procedure is used to gather user's requirements and the criteria to analyze a node performance against those requirements?

This section answers the first question and also give partial answer of the second question whereas next section will answer the second question in detail.

Numerous factors both individually or in a combined fashion can be used to evaluate the performance of a node. Sample performance factors include, network latency, response time etc. The next issue is the selection of suitable factors. This is where the second question is partially answered. Since the main goal of our approach is to allow users to customise their neighbourhoods, this decision is vested to the user to select factors according to which they want to customise their environment.

All the factors selected should be independent of each other. We can segregate the factors in two sets, factors which need to be maximized and factors which need to be minimized.

$$\text{Factors to be Maximized: } F_{Max} = \{X_1, ..., X_l\}$$
$$\text{Factors to be Minimized: } F_{Min} = \{x_1, ..., x_k\}$$

We can calculate the performance of a node using the following formula, the Norm of the node:

$$D(n) = \sqrt{\sum_{i=1}^{k}(x_{ni})^2 + \sum_{j=1}^{l}\frac{1}{(X_{nj})^2}} \tag{1}$$

To cope with the heterogeneities of scales and measurements among values of the factors, we can normalize the values to their average value.

$$D(n) = \sqrt{\sum_{i=1}^{k}\left(\frac{x_{ni}}{\bar{x}_{ni}}\right)^2 + \sum_{j=1}^{l}\left(\frac{\bar{X}_{nj}}{X_{nj}}\right)^2} \tag{2}$$

Factors can be assigned weights $\mathcal{W}$ to put emphasis on certain factors based on their importance. Above equation would become

$$D(n) = \sqrt{\sum_{i=1}^{k}\mathcal{V}_{ni}\left(\frac{x_{ni}}{\bar{x}_{ni}}\right)^2 + \sum_{j=1}^{l}\mathcal{W}_{nj}\left(\frac{\bar{X}_{nj}}{X_{nj}}\right)^2} \tag{3}$$

### 4.2 Node Radius

This section presents the detailed answer of the second question. We define the concept of a user requirement radius and its lower and upper bounds. This will be used to analyze other nodes likelihood of being selected in the neighbourhood. Besides choosing the factors, lower and upper bounds could also be set by the user:

$$\mathcal{Z}_{\mathcal{L}} = \{X_1 = \alpha_1, X_2 = \alpha_2, ..., X_n = \alpha_n\}$$
$$\mathcal{Z}_{\mathcal{U}} = \{X_1 = \beta_1, X_2 = \beta_2, ..., X_n = \beta_n\}$$

where $\{X_i, X_2, ..., X_n\}$ is a set of factors selected by the user and $\alpha_1, \alpha_2, ..., \alpha_n$ and $\beta_1, \beta_2, ..., \beta_n$ represent the lower and upper bounds for each of these factors respectively.
**Radius lower bound** is be defined to be the lower limit values of the performance requirement: $\delta_l = D(\mathcal{Z}_{\mathcal{L}})$.
**Radius upper bound** is defined in the similar way by calculating the upper value of the norm: $\delta_u = D(\mathcal{Z}_{\mathcal{U}})$.

### 4.3 Neighbourhood

Given a set of nodes N known to a node p, neighbourhood of node p denoted by Nr(P) can be defined as follows

$$\text{Nr(p)} : \forall n \in N, \delta_l \leq D(n) \leq \delta_u$$

A neighbourhood of a node will comprise all the nodes whose norm falls between the lower and upper radius of that node. The neighbourhood information will be maintained in a soft-state manner thus ensuring that stale information is erased automatically.

## 5 Distributed Search

We used distributed search methodology to process the requests issued by the users. It works in two phases as follows:

**Local phase:** The locally maintained indices of the resources in neighbourhood is searched for the availability of the resources requested by the user. Customized neighborhood will result in finding resources in this phase with a high probability.

**Forwarding Phase:** If the local phase is unsuccessful, the requests will be forwarded to nodes in the neighbourhood which can start the discovery process in a recursive manner. Contrary to P2P systems where request is either forwarded blindly or past knowledge is used while selecting neighbours to forward request, we can exploit the neighborhood connectivity pattern to use the informed forwarding strategies. Next section provides overview of two such strategies.

### 5.1 Request Forwarding

Following is the description of request forwarding strategies.

- **Request forwarding using neighbourhood:** In this request forwarding strategy, neighbours whose radius intersects the request characteristics are selected to forward the request. If no such node is found, request is forwarded to a subset of nodes. This can be a system parameter which can be set dynamically. In our experiments we set the default value to 20%.
- **Request forwarding using Routing Neighbours:** This strategy works by selecting some additional routing neighbours based on the congruence of their radius. Following is the procedure of calculating the radius congruence: Suppose a node ‘n’ is looking for routing neighbours, then for each node v ∈ N, where N is the set of nodes known to n, n will calculate the radius congruence as:

  **Lower Bound Deviation.** Lower bound deviation $\lambda_l$ is the difference in the lower bound of radius of nodes n and v.

$$\lambda_l = \begin{cases} 0 & : \ \delta_{ln} - \delta_{lv} \geq 0 \\ \delta_{lv} - \delta_{ln} & : \ otherwise \end{cases} \tag{4}$$

  **Upper Bound Deviation.** $\lambda_u$ is the difference of node v’s upper bound of radius to upper bound radius of node n.

$$\lambda_u = \begin{cases} 0 & : \ \delta_{un} - \delta_{uv} \leq 0 \\ \delta_{uv} - \delta_{un} & : \ otherwise \end{cases} \tag{5}$$

  **Radius Congruence.** The radius congruence $\sigma$ is the measurement of node v’s radius overlap with node n using the lower and upper bound deviations.

$$\sigma = \begin{cases} 0 & :(\delta_{un}-\delta_{ln}-\lambda_l-\lambda_u) \leq 0 \\ (\delta_{un}-\delta_{ln}-\lambda_l-\lambda_u) & :otherwise \end{cases} \tag{6}$$

**Normalized Radius Congruence.** $\sigma$ can be normalized to bring the value between 0 and 1.

$$\sigma_N = \frac{\sigma}{(\delta_{un} - \delta_{ln})} \tag{7}$$

Nodes are ordered according to their normalized radius congruence values and certain number of top nodes are selected. We set this number to be 5 as default.

## 6 Experimental Evaluation

### 6.1 Simulation Methodology

Simulations were performed on networks of different scales, ranging from 300 to 1500 nodes. Resource distribution to the nodes was random. Following three factors were chosen for selecting the neighbors:
**Network Latency:** Nodes were spread randomly in a 2D grid, network latency was modelled as the Euclidean distance between the coordinates.
**Processing Speed:** Processing speed is the speed of the processor at every node.
**Node Availability:** This factor models the time for which the node is available for grid usage.
Besides using the above mentioned request forwarding, a third strategy was also used for comparison. This strategy works by propagating requests to 20% random nodes. A certain percent of random nodes were chosen to start the resource discovery process. Following metrics were used to analyze the system.
**Success Rate:** Probability of requests being satisfied out of the total number of requests issued.
**Local Success Probability:** Probability of discovering resources within the local neighbourhood.
**Average Path Length:** Average number of times a request was forwarded before finding resources.
**Average # Nodes Visited:** The average number of nodes which every request have visited.
**Average # Messages Per Node:** This is the average number of requests each node in the network processes.

### 6.2 Simulation Results

**Average Path Length:** In Fig. 1a average path length is displayed. In the largest network consisting of 1500 nodes, a resource can be found in 2 hops on average. Our request forwarding strategies exploiting the neighbourhood perform quite well in all the network sizes.
**Success Rate:** Fig. 1b shows the success rate of each forwarding strategy. In a small network nodes find less nodes with congruent radius so a portion of

requests are not satisfied. As network grows bigger, nodes find more and more compatible nodes thus increasing the success rate.

**Local Success Probability:** Fig. 1c represents the probability of discovering resources within neighbourhood. Large network results in finding more suitable nodes to qualify as neighbours resulting in increased success in neighbourhood. In the largest network consisting of 1500 nodes this value is 44% which is a reasonable value but not extremely high. The reasons for not getting very high values are: 1)all the request were distinct which means a new resource was being requested. 2)Requests contained many attributes, in the case of small number of attributes especially compatible to factors already selected will result in very high rate.

**Average # nodes visited:** Graph in Fig. 1d depicts the average load on the network generated by every request issued. Flooding 20% has generated the highest load, whereas, our strategies generate a lot less network traffic.

**Average # messages per nodes:** Fig. 1e displays the other aspect of the cost of resource discovery process which is the load each network participant have to incur.

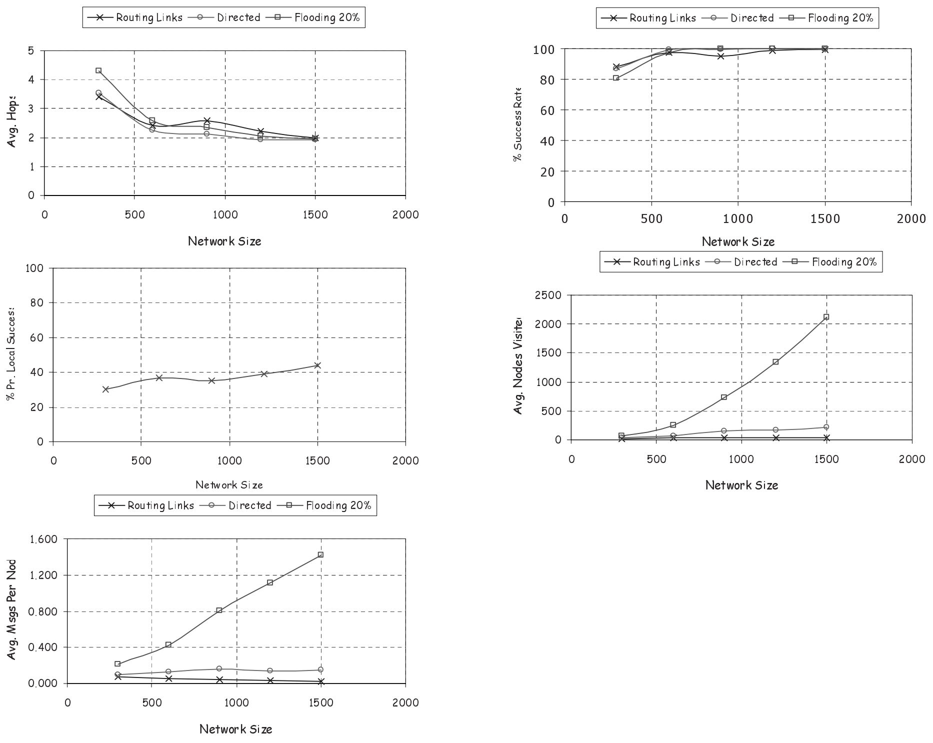

**Fig. 1.** a) Avg. Path Length b) Success Rate c) Local Success d) Avg. Nodes Visited e) Avg. Msgs Per Node

## 7 Conclusion

In this paper we have presented a new approach of discovering resources in wide area grid systems. Description of providing a customized neighborhood and distributed search was given. Simulation results were shown to prove the efficiency of our approach. Our approach works efficiently and generates less network traffic thus increasing the productivity of the system.

## References

1. Ian Foster. The anatomy of the Grid: Enabling scalable virtual organizations. *LNCS*, pages 1–??, 2001.
2. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proc. 10th IEEE Symp. On High Performance Distributed Computing*, 2001.
3. Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw. Resource management in Legion. *Future Generation Computer Systems*, pages 583–594, 1999.
4. Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of the 16th Int'l conference on Supercomputing*, 2002.
5. Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proc. of the 22nd Int'l Conference on Distributed Computing Systems*, 2002.
6. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer networks. In *Proc. of the 22nd Int'l Conference on Distributed Computing Systems*, 2002.
7. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *LNCS*, pages 329–350, 2001.
8. B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, UC Berkeley, 2001.
9. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *Proc. of ACM SIGCOMM*, 2001.
10. Ion Stoica, Robert Morris, David Karger, M. Francs Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM*, pages 149–160, 2001.
11. Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *Int'l Workshop on Grid Computing*, 2001.
12. Artur Andrzejak and Zhichen Xu. Scalable, efficient range queries for grid information services. In *Proc. of the 2nd IEEE Int'l Conference on P2P Computing*, 2002.
13. C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. In *12th IEEE Int'l Symp. on High Performance Distributed Computing*, 2003.
14. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.
15. Ok-Ki Lee and Steve Benford. An explorative model for federated trading in distributed computing environments. In *Proc. of the Int'l Conference on Open Distributed Processing*, 1995.