European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

A DHT-based Peer-to-Peer Framework for Resource Discovery in Grids

Domenico Talia, Paolo Trunfio
{talia|trunfio}@deis.unical.it
Jingdi Zeng
zeng@si.deis.unical.it
DEIS, University of Calabria,
Via Pietro Bucci 41C, 87036 Rende (CS), Italy

Mikael Högqvist
hoegquivt@zib.de
Konrad-Zuse-Zentrum für Informationstechnik Berlin,
Takustrasse 7, D-14195 Berlin-Dahlem, Germany

CoreGRID Technical Report
Number TR-0048
June 8, 2006

Institute on Knowledge and Data Management &
Institute on System Architecture

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

A DHT-based Peer-to-Peer Framework for Resource Discovery in Grids

Domenico Talia, Paolo Trunfio
{talia|trunfio}@deis.unical.it
Jingdi Zeng
zeng@si.deis.unical.it
DEIS, University of Calabria,
Via Pietro Bucci 41C, 87036 Rende (CS), Italy

Mikael Högqvist
hoegquivt@zib.de
Konrad-Zuse-Zentrum für Informationstechnik Berlin,
Takustrasse 7, D-14195 Berlin-Dahlem, Germany

CoreGRID TR-0048

June 8, 2006

Abstract

Several systems adopting Peer-to-Peer (P2P) solutions for resource discovery in Grids have recently been proposed. This report looks at a P2P resource discovery framework aiming to manage various Grid resources and complex queries. Following the discussion on characteristics of Grid resources and related query requirements, a DHT-based framework leveraging different P2P resource discovery techniques is proposed. The goal of the proposed framework is two-fold: to address discovery of multiple resources, and to support discovery of dynamic resources and arbitrary queries in Grids.

1 Introduction

A large amount of work on Peer-to-Peer (P2P) resource discovery has been done, including both unstructured and structured systems. Early unstructured P2P systems, such as Gnutella[1], use the *flooding technique* to broadcast resource requests in the network. The flooding technique does not rely on a specific network topology and supports queries in arbitrary forms. Several approaches [2, 3, 4], moreover, have been proposed to enhance two intrinsic drawbacks of the flooding technique: the potentially massive amount of messages, and the possibility that an existing resource may not be located. For structured P2P systems, distributed hash tables (DHTs) are widely used. *DHT-based systems* [5, 6, 8] arrange $< attribute, attribute\text{-}value >$, that is, $< key, value >$ pairs in multiple locations across the network. A query message is forwarded towards the node that is responsible for the key in a limited number of hops. The result is guaranteed, if such a key exists in the system. As compared to the flooding technique, however, DHT-based approaches need intensive maintenance on hash table updates.

In Grid environments, applications are composed of hardware and software resources that need to be located; efficient and effective resource discovery is critical. While Grids can be seen as a more federated version of P2P systems, they shall be able to benefit from distributive P2P resource discovery techniques. Taking into account the characteristics of Grids, accordingly, several P2P resource discovery techniques have been adapted to Grid environments. For

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

instance, DHT-based P2P resource discovery systems have been extended to support range value and multi-attribute queries [9, 10, 11, 12, 13].

Two major differences between P2P systems and Grids, however, determine their different approaches towards resource discovery. First, P2P systems are originally designed to share files among peers. Grids, on the contrary, deal with a complicate set of resources, ranging from files to computing resources. Second, the dynamism of P2P systems comes from both nodes and resources. Peers join and leave at any time, and thus do the resources shared in the network. In Grid environments, nodes connect to the network in a relatively static manner. The dynamism of Grids mainly comes from the fast-changing status of resources. For example, the storage space and CPU load may change continuously over time.

Highlighting the variety and dynamism of Grid resources, this report proposes a DHT-based resource discovery framework for Grids. The rest of the report is organized as follows. Section 2 introduces existing Grid resource discovery systems that relate to this work. Section 3 discusses characteristics of Grid resources and related query requirements. Section 4 unfolds the picture of the proposed framework, by touching up on major implementation concerns. Section 5 concludes the report.

2 Related Work

Several systems exploiting DHT-based P2P approaches for resource discovery in Grids have recently been proposed [9, 10, 11, 12, 13]. Two important issues investigated by these systems are range queries and multi-attribute resource discovery.

Range queries look for resources specified by a range of attribute values (e.g., a CPU with speed from 1.2GHz to 3.2GHz). These queries are not supported by standard DHT-based systems such as Chord [5], CAN [6], and Pastry [7]. To support range queries over DHTs, a typical approach is to use locality preserving hashing functions that retain the order of numerical values in DHTs [9, 10].

Multi-attribute resource discovery refers to the problem of locating resources that are described by a set of attributes or characteristics (e.g., OS version, CPU speed, etc.). Several approaches have been proposed to organize resources in order to efficiently support multi-attribute queries. Some systems focus on weaving all attributes into one DHT [11] or one tree [12]. Some others adopt one DHT for each attribute [9, 10, 13].

Aside from single value queries, range queries, and multi-attribute queries for single resources, the proposed framework aims to support queries for multiple resources. We use multiple DHTs to manage attributes of multiple resources. This provides a straightforward architecture, and leaves space for potential extensions.

Gnutella-based *dynamic query* [14] strategy is used to reduce the number of messages generated by flooding. Instead of all directions, this strategy forwards the query only to a selected peer. If a response is not returned from a direction, another round of search is initiated in the next direction, after an estimated time. For relatively popular contents this strategy significantly reduces the number of messages without increasing the response time.

Broadcast in DHT-based P2P networks [15] adds broadcast service to a class of DHT systems that have logarithmic performance bounds. In a network of N nodes, the node that starts the broadcast reaches all other nodes with exactly $N - 1$ messages (i. e., no redundant messages are generated).

The approach proposed for dynamic resource discovery in this report uses a DHT for broadcasting queries to all nodes without redundant messages, and adopts a similar "incremental" approach of dynamic query. It reduces the number of exchanged messages and response time.

3 Background

Two concepts are used in the rest of the report:

- *Resource class*: a "model" for representing resources of the same type. Each resource class is defined by a set of attributes which specify its characteristics.
- *Resource*: an "instance" of a resource class. Each resource has a specific value for each attribute defined by the corresponding resource class. Resources are univocally identified by URLs.

An example of resource class is "computing element" that defines the common characteristics of computing resources. These characteristics are described by attributes such as "OS name", "CPU speed", and "Free memory".

A computing resource (i.e., an instance of the "computing element" resource class) has a specific value for each attribute, for example, "OS name = Linux", "CPU speed = 1000MHz", and "Free memory = 1024MB".

Table 1 lists some examples of Grid resources classes. It's not exhaustive, with new classes emerging. A more complete list of resource classes can be found in [18].

Resource class	**Description**
Computing resource	Computing capabilities provided by computers, clusters of computers, etc.
Storage resource	Storage space such as disks, external memory, etc.
Network resource	Network connections that ensures collaboration between other Grid resources.
Device resource	Specific devices such as instruments, sensors, etc.
Software resource	Operating systems, software packages, Web services, etc.
Data resource	Various kinds of data stored in file systems or databases.

Table 1: Examples of Grid resource classes.

Resource classes can be broadly classified into *intra-node* and *inter-node* resources. "Computing element" is an example of intra-node resource class. An example of inter-node resource class is "network connection" (see Table 1), which defines network resource characteristics.

Figure 1 shows a simple Grid including four nodes and three resource classes. As examples of intra-node resources, $NodeA$ includes two instances of resource class *a* and one instance of resource class *b*. The figure also shows two inter-node resources: one between $NodeA$ and $NodeB$, and the other between $NodeB$ and $NodeD$.

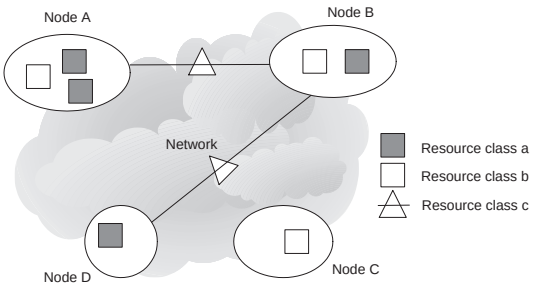

Figure 1: Inter-node and intra-nodes resources.

The attributes of each resource class are either *static* or *dynamic*:

- Static attributes refer to resource characteristics that do not change frequently, such as "OS name" and "CPU speed" of a computing resource.
- Dynamic attributes are associated to fast changing characteristics, such as "CPU load" and "Free memory".

The goal of resource discovery in Grids is to locate resources that satisfy a given set of requirements on their attribute values.

Three types of queries apply to each attribute involved in resource discovery:

- *Exact match query*, where attribute values of numeric, boolean, or string types are searched.
- *Range query*, where a range of numeric or string values are searched.
- *Arbitrary query*, where for instance partial phrase match or semantic search is carried out.

A multi-attribute query is composed of a set of sub-queries on single attributes. Each sub-query fits in one of the three types as listed above, and the involved attributes are either static or dynamic.

Complex Grid applications involve multiple resources. Thus, multi-resource queries are often needed. For instance, one can be interested in discovering two computing resources and one storage resource; these resources may

not be geographically close to each other. A multi-resource query, in fact, involves a set of sub-queries on individual resources, where each sub-query can be a multi-attribute query.

Taking into consideration both characteristics and query requirements of Grid resources, appropriate P2P discovery techniques are listed in Table 2. DHTs are used for exact and range queries on static Grid resources, where static resources are quantified and represented by exact numeric values. Organized in DHTs, these values can be efficiently accessed. Nevertheless, traditional DHTs may not be able to handle fast-changing attribute values. In other words, the frequent updates on DHTs that cause inconsistent table content and unstable state pose a challenge on the resource discovery performance of DHTs. Moreover, queries in arbitrary forms are not supported by DHTs. As a result, the flooding technique is used for both dynamic Grid resources and arbitrary queries on static Grid resources.

	Static Grid resources	**Dynamic Grid resources**
Exact query	DHT	Flooding
Range query	DHT	Flooding
Arbitrary query	Flooding	Flooding

Table 2: Query techniques used for different types of resources and queries.

4 System architecture

The framework aims to provide a generic architecture that leverages existing techniques to fulfill various resource discovery needs in Grid environments. As illustrated in Figure 2, the DHT-based architecture is composed of a set of *virtual planes*, one for each resource class. Within the virtual plane of resource class R_a, for example, static attributes $R_a.A_{s1}$, ..., $R_a.A_{sn}$ are associated to their DHTs, respectively. Exact or range queries on *static* attributes are carried out using the DHTs corresponding to these attributes.

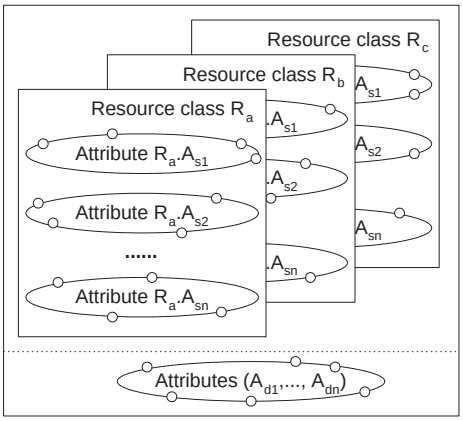

Figure 2: System architecture.

An additional "general purpose" DHT is dedicated to queries on dynamic attributes and to "arbitrary queries" on static attributes. This DHT is different from the others. The DHTs in the virtual planes are standard DHTs, in which both nodes and resource identifiers are mapped on the same ring. In the general purpose DHT, only the node identifiers are mapped to the ring, while resources are not mapped to it. In other terms, there are not pointers to resources in the general purpose DHT.

Basically, the general purpose DHT is used to broadcast queries to all Grid nodes whose identifiers are mapped to the ring. All Grid nodes reached by a query are in charge of processing it against the local resources, and sending the response to the node that originated the query. The mechanism used for broadcasting a query on this ring is described in Section 4.3.

4.1 Local component

Figure 3 shows the software modules inside each Grid node.

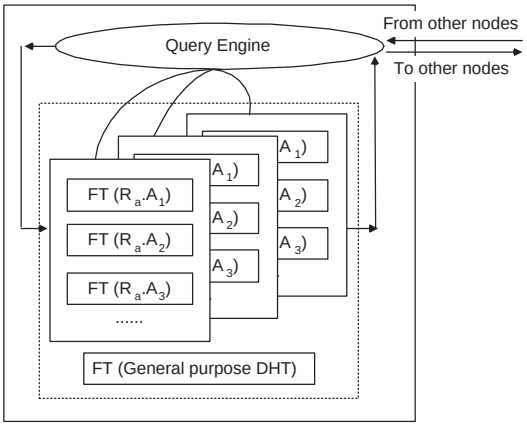

Figure 3: Software modules inside each Grid node.

With multiple virtual planes defined in the system, each node participates in all DHTs of these virtual planes. Therefore, multiple finger tables corresponding to each DHT co-exist in each node, as illustrated in Figure 3. For example, finger tables $FT(R_a.A_1)$, $FT(R_a.A_2)$,..., and $FT(R_a.A_n)$ correspond to DHTs of attributes $R_a.A_{s1}$...$R_a.A_{sn}$ in Figure 2.

The finger table of the general purpose DHT, that is, $FT(General\,purpose\,DHT)$, is used to reach individual nodes and locate dynamic attributes A_{d1},...,A_{dn}. A query engine processes resource discovery requests and associates them to different query instances and thus DHTs. The results are then generated at the node where related queries are initiated.

4.2 Static attribute discovery

A number of multi-attribute, range query approaches has emerged. They either use one DHT [11] or one tree [13] for all attributes, or arrange attribute values on multiple DHTs [9]. While both one DHT and multi-DHT approaches have proved effective, we adopt the multi-DHT strategy, for its simplicity and extension potentials.

Assume there are p classes of resources, each of which has q types of attributes. A total number of $p \times q$ DHTs is used. Although one node does not necessarily have all attributes, it is included in all DHTs, and the values of its blank entries are left as null. The number of finger tables that a node maintains is $p \times q$. Each finger table, for overall N nodes, contains $log(N)$ entries.

While existing approaches support resource discovery on single or multiple attributes of one resource class, the framework proposed in this report manages multiple resources. One way to do it is to hash the string of "resource class + attribute" into an *ID*; this ID is used to identify the corresponding finger table inside a node.

4.3 Dynamic Attribute Discovery

As mentioned in Section 2, our approach for dynamic resource discovery exploits both the dynamic query [14] and the broadcast over DHT [15] strategies. The general purpose DHT and associated finger tables, as illustrated in Figures 2 and 3, are used only to index Grid nodes, without keeping pointers to Grid resource attributes. Queries are then processed by the local query engine of each node.

4.3.1 To Reach all Nodes

To reach all nodes without redundant messages, the broadcast strategy is based on a DHT [15]. Take a fully populated Chord ring with $N = 2^M$ nodes and a M-bit identifier space as an example. Each Chord node k has a finger table, with fingers pointing to nodes $k + 2^{i-1}$, where $i = 1, ..., M$. Each of these M nodes, in turn, has its fingers pointing to another M nodes. Each node forwards the query to all nodes in its finger table, and in turn, these nodes do the same

with nodes in their finger tables. In this way, all nodes are reached in M steps. Since multiple fingers may point to the same node, a strategy is used to avoid redundant messages. Each message contains a "limit" argument, which is used to restrict the forwarding space of a receiving node. The "limit" argument for the node pointed by finger i is finger $i+1$.

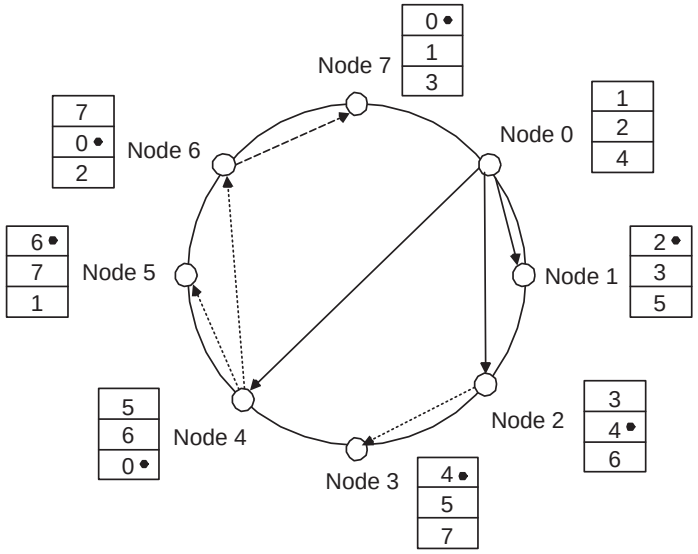

Figure 4: An example of broadcast.

Figure 4 gives an example of an eight-node three-bit identifier Chord ring. The "limit" of each node is marked with a black dot. Three steps of communication between nodes are demonstrated with solid, dotted, and dashed lines. Obviously, node 0 reaches all other nodes via $N-1$ messages within M steps. The same procedure applies to Chord ring with $N < 2^M$ (i.e., not fully populated networks). In this case, the number of distinct fingers of each node is $logN$ on the average.

4.3.2 Incremental Resource Discovery

The broadcast strategy in Section 4.3.1 adopts a "parallel" approach. That is, the node that initiates the discovery tasks sends the query message to all its fingers in parallel. Although no redundant messages are generated in the network, its $N-1$ messages can be prohibitive in large-scale Grids.

Referred to as "incremental", our approach uses a mixed parallel and sequential query message forwarding. A "parallel degree" D is introduced to adjust the range of parallel message forwarding, and thus curb the number of exchanged messages. Given a node that initiates the query, it forwards the query message in parallel to nodes pointed by its first D distinct fingers. If there are enough positive responses, the search terminates; otherwise, this node forwards the query message to the node pointed by its $D+1$ finger. The same procedure applies to nodes pointed by the rest of fingers, sequentially, until the number of positive responses meets the requirements. When $D = M$, our incremental approach is the same as the parallel approach; when $D = 1$, the incremental approach proceeds in a completely sequential manner, where nodes pointed by all fingers are visited one after another.

The number of generated messages by the incremental approach is obviously less than or equal to that of the parallel approach. The response time of the incremental approach, however, may be prolonged owing to its sequential query message forwarding. We argue that this does not necessarily hold true. In large-scale Grids, multiple query requests at one node can be prominent, which adds extra delay to response time. Under this circumstance, the incremental approach shall benefit from its reduced number of messages that may shorten this extra delay.

A discrete-event simulator is used to evaluate the performance of the incremental approach in comparison with the parallel approach. Preliminary results confirmed the previous expectation. When the number of queries concurrently submitted in the network increases, the response time of the incremental approach increases more slowly than that of the parallel approach. This is because in the parallel approach the overall number of generated messages is much higher than that in the incremental approach, which results in increased message traffic and processing load that cause

a higher response time. In the scenario where the processing time and the delivery time increase exponentially with the load of the network, the response time in the incremental approach should be significantly better than that in the parallel approach.

4.3.3 Discussion on tradeoffs

The two major motivations of the "incremental" approach are to support arbitrary queries and to acquire up-to-date values of dynamic resource attributes.

Arbitrary queries. The term "arbitrary query" is used in this report to indicate any query that cannot be solved through key lookup over a *standard* DHT. Examples of arbitrary queries include partial-match queries (e.g., substrings searching), and queries against structured and semi-structured data stored in different formats across Grid nodes.

Extensions to standard DHTs have been proposed to support keyword searching [20], complex queries [21], and efficient storing and querying of XML data (e.g., [22] for RDF documents). However, such systems require additional indexing structures, which may result ineffective in case of highly-dynamic resources. Moreover, the use of heterogeneous resource schemas in different sites makes infeasible the design of ad hoc indexing structures that are able to support queries of arbitrary format.

For these reasons, following the approach in [15], the broadcast over DHT approach is adopted to distribute arbitrary queries across the Grid. This allows to search resources of interest even if they cannot be indexed by a dedicated infrastructure. To compensate the inherent broadcast overhead, the "incremental" strategy is used, reducing response time and saving network bandwidth in case of relatively popular resources, as discussed before.

Up-to-date attribute values. The "incremental" approach is beneficial when DHT updating is rather frequent, and thus up-to-date attribute values are difficult to obtain. A Grid user discovers desired resources, and negotiates with them to ensure the eventual resource usage. During the course of the negotiation, however, the targeted resource attributes may change. This inconsistency of attribute values will cost the Grid user one or more rounds of query and negotiation.

Additionally, when using the "incremental" approach, there is no message overhead for updating the dynamic resource attribute values. The saved cost increases with the updating frequency, the number of attributes, and the DHT updating cost of a single value, i.e., O(log N).

Given the advantages and drawbacks of different strategies, the Grid resource provider can decide if an attribute should be treated as static or dynamic. Accordingly, the multi-DHT or the broadcast over DHT approach is used.

4.4 APIs

DHT systems have APIs developed to maintain their hash tables. APIs of Chord, for example, include *insert(key, value)*, *lookup(key)*, *update(key, newval)*, *join(n)*, and *leave()*. Systems [15] built on top of DHT systems also define policies and higher level APIs.

The framework proposed in this report identifies two types of API operations: *local* operations retrieve and update attribute states; *communication* operations define the messages exchanged among nodes, and the information carried by these messages. Extra APIs to differentiate multiple resource classes is of importance here. For example, *IDs* are used to identify attributes of different resource classes. These *IDs* may be arranged in a list or a DHT.

There are two approaches towards the definition of a complete set of APIs for the proposed framework. First, new higher level APIs are introduced to interface with existing APIs. Second, existing APIs are modified to cope with attributes of multiple resource classes.

To eliminate the difference between structured overlay protocols (e.g., CAN, Tapstry, Pastry, and Chord), basic abstractions for API implementation in structured P2P networks are proposed [19]. The proposal demonstrates how higher level abstractions, such as dynamic hash table operations, are realized on top of a basic key-based routing layer. This could be an interesting direction for the API implementation of the framework.

5 Conclusions

The proposed framework aims to support resource discovery in Grid environments, based on DHT systems. Aside from the proposed framework, this report identifies the necessity of handling multiple Grid resources and dynamic

Grid attributes. Methods that support resource discovery on multiple resource classes as well as on dynamic attributes and arbitrary queries are introduced.

To be implemented over different platforms and span through wide area networks, this framework needs to be seamlessly integrated with the *Open Grid Services Architecture* (*OGSA*) and the *Web Service Resource Framework* (*WSRF*), which provides standard mechanisms for the implementation of Grid systems and applications [17]. Moreover, the framework needs to explicitly address XML-based queries to support the discovery of WSRF-enabled Web services.

Acknowledgments

We thank Dr. Artur Andrzejak for reading and commenting on an earlier version of this report.

References

[1] Gnutella Protocol Development.
http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[2] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid Search Schemes for Unstructured Peer-to-peer Networks," Proc. of IEEE INFOCOM'05, Miami, USA, March 2005.

[3] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replicating in Unstructured Peer-to-peer Networks," Proc. of 16th Annual ACM Int. Conf. on Supercomputing (ISC'02), New York, USA, June 2002.

[4] A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-peer Systems," Proc. of Int. Conf. on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2002.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," Proc. of ACM SIGCOMM'01, San Diego, USA, August 2001.

[6] S. Ratnasany, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. of ACM SIGCOMM'01, San Diego, USA, August 2001.

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pp. 329-350, November 2001.

[8] M. Frans Kasshoek and D. R. Karger, "Koorde: A Simple Degree-optimal Distributed Hash Table," Proc. of 2nd Int. Workshop on Peer-to-peer Systems (IPTPS'03), Berkeley, USA, February 2003.

[9] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," Journal of Grid Computing, 2004.

[10] A. Andrzejak and Z. Xu, "Scalable, Efficient Range Queries for Grid Information Services," Proc. of 2nd IEEE Int. Conf. on Peer-to-peer Computing (P2P'02), Sweden, September 2002.

[11] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Scalable Wide-Area Resource Discovery," UC Berkeley Technical Report, UCB/CSD-04-1334, July 2004.

[12] S. Basu, S. Banerjee, P. Sharma, S. Lee, "NodeWiz: Peer-to-peer Resource Discovery for Grids," Proc. of IEEE/ACM GP2PC'05, Cardiff, UK, May 2005.

[13] D. Spence and T. Harris, "XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform," Proc. of HPDC'03, Washington, USA, June 2003.

[14] Adam A. Fisk, "Gnutella Dynamic Query Protocol v0.1," http://www.the-gdf.org/wiki/index.php?title=Dynamic_Querying.

[15] S. El-Ansary, L. Alima, P. Brand, and S. Haridi, “Efficient Broadcast in Structured P2P Networks” Proc. of IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGRID’05), Cardiff, UK, 2005.

[16] J. Salter, “An Efficient Reactive Model for Resource Discovery in DHT-based Peer-to-peer Networks,” Master Thesis, University of Surrey, UK, June 2005.

[17] C. Comito, D. Talia, and P. Trunfio, “Grid Services: Principles, Implementations and Use,” International Journal of Web and Grid Services, Vol. 1, No. 1, pp 48-68, 2005.

[18] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Konya, M. Mambelli, J. Schopf, M. Viljoen, and A. Wilson, “GLUE Schema Specification Version 1.2: Final Specification - 3 Dec 05,” http://infnforge.cnaf.infn.it/glueinfomodel/index.php/Spec/V12.

[19] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, “Towards a Common API for Structured Peer-to-peer Overlays”, Proc. of IPTPS’03, Berkeley, USA, February 2003.

[20] P. Reynolds and A. Vahdat, “Efficient Peer-to-Peer Keyword Searching”, Proc. of Int. Middleware Conference (Middleware 2003), Rio de Janeiro, Brazil, 2003.

[21] M. Harren, J. M. Hellerstein, R. Huebsch, and B. T. Loo, “Complex Queries in DHT-based Peer-to-Peer Networks”, Proc. of 1st Int. Workshop on Peer-to-Peer Systems (IPTPS’02), 2002.

[22] M. Cai, M. R. Frank, B. Yan, and R. M. MacGregor, “A Subscribable Peer-to-Peer RDF Repository for Distributed Metadata Management”, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2(2):109.130, December 2004.