Peer-to-Peer Models for Resource Discovery in Large-scale Grids: A Scalable Architecture

Domenico Talia[1,2], Paolo Trunfio[1,2], and Jingdi Zeng[1,2]

[1] DEIS, University of Calabria
Via P. Bucci 41c, 87036 Rende (CS), Italy
[2] CoreGRID NoE
{talia, trunfio, zeng}@si.deis.unical.it

Abstract. As Grids enlarge their boundaries and users, some of their functions should be decentralized to avoid bottlenecks and guarantee scalability. A way to provide Grid scalability is to adopt *Peer-to-Peer* (*P2P*) models to implement non hierarchical decentralized Grid services and systems. A core Grid functionality that can be effectively redesigned using the P2P approach is *resource discovery*. This paper proposes a P2P resource discovery architecture aiming to manage various Grid resources and complex queries. Its goal is two-fold: to address discovery of multiple resources, and to support discovery of dynamic resources and arbitrary queries in Grids. The architecture includes a scalable technique for locating dynamic resources in large-scale Grids. Simulation results are provided to demonstrate the efficiency of the proposed technique.

1 Introduction

In Grid environments, applications are composed of dispersed hardware and software resources that need to be located and remotely accessed. Efficient and effective resource discovery is then critical. Peer-to-Peer (P2P) techniques have been recently exploited to achieve this goal.

A large amount of work on 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 the resource requests in the network. The flooding technique does not rely on a specific network topology and supports queries in arbitrary forms. Several approaches [2–4], moreover, have been proposed to solve two intrinsic drawbacks of the flooding technique, i.e., the potentially massive amount of messages, and the possibility that an existing resource may not be located. In structured P2P networks, Distributed Hash Tables (DHTs) are widely used. *DHT-based systems* [5–7] arrange $< key, value >$ 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 unstructured systems, however, DHT-based approaches need intensive maintenance on hash table updates.

Taking into account the characteristics of Grids, several P2P resource discovery techniques have been adapted to such environments. For instance, DHT-based P2P resource discovery systems have been extended to support range value and multi-attribute queries [8–11]. Two major differences between P2P systems and Grids, however, determine their different approaches towards resource discovery. First, P2P systems are typically designed to share files among peers. Differently, Grids deal with a set of different 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 among them. In Grid environments, nodes connect to the network in a relatively more stable manner. The dynamism of Grids mainly comes from the fast-changing statuses of resources. For example, the storage space and CPU load can change continuously over time.

Highlighting the variety and dynamism of Grid resources, this paper proposes a DHT-based resource discovery architecture for Grids. The rest of the paper is organized as follows. Section 2 introduces existing Grid resource discovery systems that relate to our work. Section 3 discusses characteristics of Grid resources and related query requirements. Section 4 unfolds the picture of the proposed architecture, and studies the performance of its dynamic resource discovery strategy through simulations. Section 5 concludes the paper.

2 Related Work

Several systems exploiting DHT-based P2P approaches for resource discovery in Grids have recently been proposed [8–10]. 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, a typical approach is to use locality preserving hashing functions, which retain the order of numerical values in DHTs [8, 9].

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 [10] or one tree [12]. Some others adopt one DHT for each attribute [9, 11].

Aside from single value queries, range queries, and multi-attribute queries for single resources, the proposed architecture 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* [13] 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 [14] 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 paper is inspired by both the dynamic query strategy and the broadcast approach mentioned above. It uses a DHT for broadcasting queries to all nodes without redundant messages, and adopts a similar "incremental" approach of dynamic query. This approach reduces the number of exchanged messages and response time, which ensures scalability in large-scale Grids.

3 Resources and Query Types

In Grids, *resources* belong to different *resource classes.* A *resource class* is a "model" for representing resources of the same type. Each resource class is defined by a set of attributes which specify its characteristics. A *resource* is 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 resource" that defines the common characteristics of computing resources. These characteristics are described by attributes such as "OS name", "CPU speed", and "Free memory". An instance of the "computing 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. A more complete list of resource classes can be found in [15].

Table 1. Examples of Grid resource classes.

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 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.

Resource classes can be broadly classified into *intra-node* and *inter-node* resources. "Computing resource" 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. Fig. 1 shows a simple Grid including four nodes and three resource classes. As examples of inter-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 $NodeD$, and the other between $NodeB$ and $NodeD$.

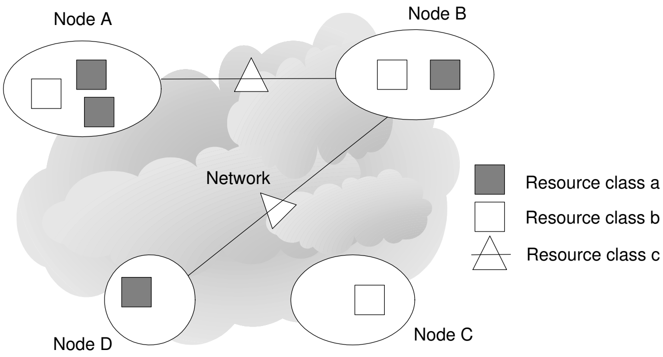

Fig. 1. Inter-node and intra-nodes resources.

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 is 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, the appropriate P2P discovery techniques are listed in Table 2.

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

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

As shown in the table, DHTs are used only for exact and range queries on static Grid resources. For dynamic resources and arbitrary queries, classical DHT-based approaches are not suitable. These approaches were not originally designed for resource discovery queries of arbitrary expression forms. Moreover, fast-changing resources, such as CPU load, require frequent updates on DHTs, and thus cause prohibitive maintenance costs. Flooding approaches are used for both dynamic Grid resources and arbitrary queries on static resources. This is because flooding does not require table updates and maintenance. Nevertheless, the massive amount of messages they generate do not scale with network sizes.

4 System Architecture

The framework aims to provide a generic architecture that leverages existing techniques to fulfill various resource discovery needs in Grid environments. In order to exploit diverse resource discovery techniques, the DHT-based architecture described in Fig. 2 is proposed.

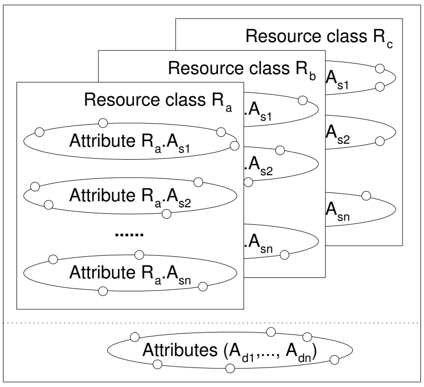

Fig. 2. System architecture.

The system 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.

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 DHTs in the virtual planes. The DHTs in the virtual plane are standard DHTs, in which both nodes and resource identifiers are mapped to the same ring. In general purpose DHT, only node identifiers are mapped to the ring, while resources are not mapped to it. In other words, there are not pointers to resources in the general purpose ring.

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 initiated the query. The mechanisms used for broadcasting a query on this ring are described in Section 4.3.

4.1 Local Component

Fig. 3 shows the 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 Fig. 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 Fig. 2.

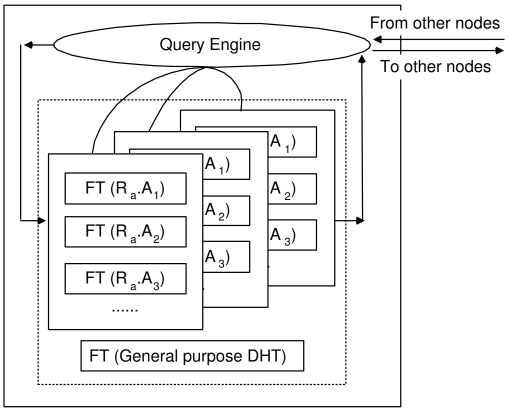

Fig. 3. Software modules inside each Grid node. FTs are finger tables associated to the used DHTs.

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 have emerged. They either use one DHT [10] or one tree [11] for all attributes, or arrange attribute values on multiple DHTs [9]. While both single-DHT and multi-DHT approaches have proved effective, we adopt the multi-DHT strategy because of its simplicity and extension potentials.

Assume there are p classes of resources, each of which has q types of attributes. 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 $p \times q$.

While existing approaches support resource discovery on single or multiple attributes of one resource class, the architecture proposed in this paper manages multiple resources. One way to do this is to hash the string of "resource class + attribute" into a *DHT 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 [13] and the broadcast over DHT [14] strategies. The general purpose DHT and associated finger tables, as illustrated in Figs. 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.

To Reach all Nodes. To reach all nodes without redundant messages, the broadcast strategy is based on a DHT [14]. Taking 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 of a message for the node pointed by finger i is finger $i + 1$.

Fig. 4 gives an example of an eight-node three-bit identifier Chord ring. The limit of broadcast is marked with a black dot. Three steps of communication between nodes are demonstrated with solid, dotted, and dashed lines. Obviously,

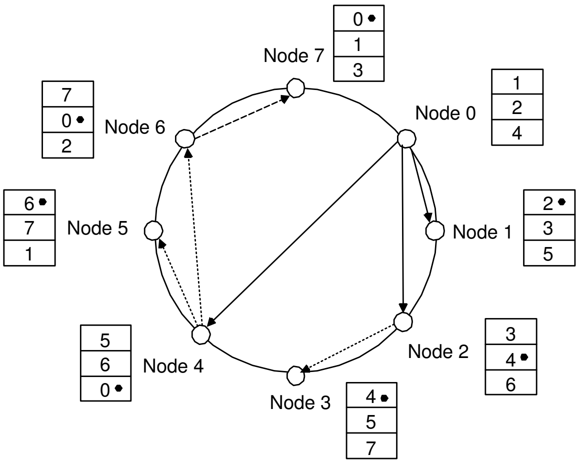

Fig. 4. An example of broadcast.

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.

Incremental Resource Discovery. The broadcast over DHT presented above 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 distinct D fingers. If there is a positive response, the search terminates; otherwise, this node forwards the query message to the node pointed by its $D + 1$ finger. This procedure applies to nodes pointed by the rest of fingers, sequentially, until a positive response returns.

When $D = M$, our incremental approach turns into the parallel one; when $D = 1$, the incremental approach becomes a sequential one, 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 one. The response time of 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 shortens this extra delay.

Performance Evaluation. A discrete-event simulator has been implemented to evaluate the performance of the incremental approach in comparison with the parallel approach.

Two performance parameters have been evaluated: the *number of messages* Q and the *response time* T. Q is the total number of exchanged messages in the network, and T is the time a node waited to receive the first response (i.e., the first query hit).

The system parameters are explained in Table 3. In all simulations we used $M = 32$ and $D = 7$. The number of nodes N ranges from 2000 to 10000, R ranges from 10 to 1000, and P ranges from 0.005 to 0.25. The values of performance parameters are obtained by averaging the results of 10 to 30 independent simulation runs.

Table 3. System parameters.

Parameter	**Description**
M	Number of bits of node identifiers.
N	Number of Grid nodes in the network.
R	Number of nodes that concurrently submit query requests.
P	Fraction of nodes in the network that possesses the desired resource.
D	Number of first distinct fingers the search is conducted on in parallel.

The time to pass a message from $NodeA$ to $NodeB$ is calculated as the sum of a processing time and a delivery time. The processing time is proportional to the number of queued messages in $NodeA$, while the delivery time is proportional to the number of incoming messages at $NodeB$. In this way, the response time depends on both message traffic and processing load of nodes.

Table 4 shows the number of exchanged messages in both parallel and incremental strategies, with $R = 1$. The parallel strategy always generates $N - 1$ messages for each submitted query, which could be prohibitive for large-scale Grids. In the incremental approach, the number of messages is dramatically reduced. Moreover, when the value of P is over a certain limit, the number of messages fluctuates around the value of 2^D and it does not depend from the number of nodes (i.e., network size). This limit is determined by the number of Grid nodes N, the fraction of nodes with matching resources P, and the number of first distinct fingers D.

For example, in a network with $N = 10000$, when $P = 0.1$ the number of matching resources is $N \times P = 1000$. The number of nodes included in the first $D = 7$ fingers is $2^D = 128$, on the average. Obviously, this density is high enough for the incremental strategy to locate the desired resource within the first D fingers. With a lower value of P, nevertheless, the search needs to go beyond the first D fingers; this introduces a fluctuation in the number of exchanged messages, as in the case of $P = 0.005$.

Table 4. Comparison on the number of exchanged messages (Q) in parallel and incremental approaches.

N	P	Q (Parallel)	Q (Incremental)
2000	0.005	1999	279
	0.10	1999	127
	0.25	1999	126
4000	0.005	3999	326
	0.10	3999	129
	0.25	3999	124
6000	0.005	5999	291
	0.10	5999	126
	0.25	5999	126
8000	0.005	7999	282
	0.10	7999	128
	0.25	7999	128
10000	0.005	9999	389
	0.10	9999	127
	0.25	9999	125

Figs. 5, 6 and 7 show the response time in networks composed by 2000, 6000 and 10000 nodes, respectively, with $P = 0.10$ and values of R ranging from 10 to 1000. The response time is expressed in time units.

The main result shown in Figs. 5, 6 and 7 is that, for any value of N, when the values of R are at the lower end of its range, the parallel approach has a shorter response time. When the value of R increases, the incremental approach outperforms the parallel one. This is because in the parallel approach the overall number of generated messages is much higher than the one in the incremental approach, resulting in increased message traffic and processing load that cause a higher response time.

It can be also noted that, for any value of R, the response time decreases as N increases. This trend is similar in both parallel and incremental approaches. This is because the probability of finding the desired resource in a given time interval is proportional to the number of nodes that possess it, and this in turn is proportional to the network size.

The simulation results demonstrate that with higher values of R the incremental approach scale better than the parallel one. It is important to recall that in our simulator the processing time is proportional to the number of messages to be processed, and the delivery time is proportional to the number of messages to be delivered. Therefore, the response time increases linearly with message traffic and load of nodes. In a more realistic scenario the processing time and the delivery time may increase exponentially with the load of the network. In this case, the response time in the incremental approach would result significantly better that the parallel one. To better evaluate the effect of high loads in

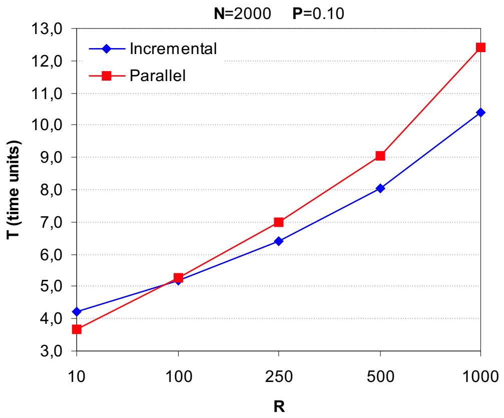

Fig. 5. Response time of parallel and incremental approaches ($N = 2000$).

large-scale Grids, we are currently studying the use of more complex processing and delivery time functions in our simulator.

5 Conclusions

This paper discussed the characteristics of Grid resources and identified critical problems of resource discovery in Grids. A DHT-based P2P framework has been introduced to address the variety and dynamism of Grid resources. It exploits multiple DHT and existing P2P techniques for multiple static resources and implements an “incremental” resource discovery approach for dynamic resources. As compared to the original strategy, the incremental approach generates reduced number of messages and experiences lower response time in large-scale Grids.

With the emergence of service-oriented Grids [16], *service discovery* has become an important topic. Grid services are today implemented complying with the *Web Services Resource Framework* (*WSRF*) family of specifications, which define standard mechanisms for accessing and managing Grid resources using *Web services*. Web services are defined using XML-based languages, and XML queries are used to query their features. As a future work, the architecture in this paper could be extended to address Grid service discovery, in particular, dynamic service indexing and XML-based queries support.

Acknowledgements

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

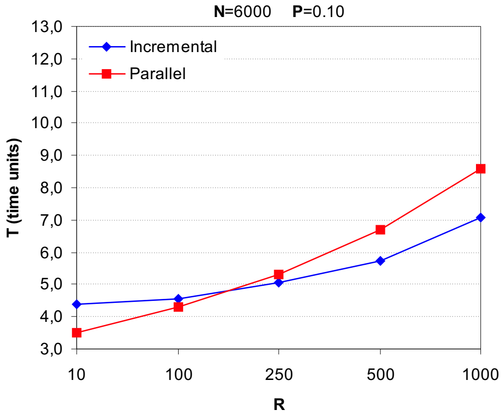

Fig. 6. Response time of parallel and incremental approaches ($N = 6000$).

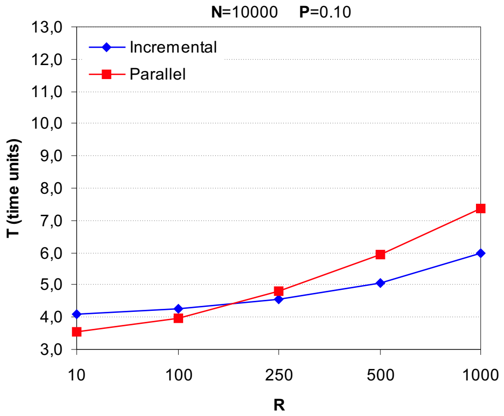

Fig. 7. Response time of parallel and incremental approaches ($N = 10000$).

work has been also supported by the Italian MIUR FIRB Grid.it project RBNE01KNFP on High Performance Grid Platforms and Tools.

References

1. Gnutella Protocol Development. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.
2. Gkantsidis, C., Mihail, M., Saberi, A.: Hybrid Search Schemes for Unstructured Peer-to-peer Networks. Proc. of IEEE INFOCOM'05, Miami, USA (2005).

3. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replicating in Unstructured Peer-to-peer Networks. Proc. of 16th Annual ACM Int. Conf. on Supercomputing (ISC'02), New York, USA (2002).
4. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-peer Systems. Proc. of Int. Conf. on Distributed Computing Systems (ICDCS'02), Vienna, Austria (2002).
5. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Proc. of ACM SIGCOMM'01, San Diego, USA (2001).
6. Ratnasany, S., Francis, P., Handley, M., Karp, R. M., Shenker, S.: A Scalable Content-Addressable Network. Proc. of ACM SIGCOMM'01, San Diego, USA (2001).
7. Rowstron, A., Druschel, P.: 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 (2001).
8. 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), Linköping, Sweden (2002).
9. Cai, M., Frank, M., Chen, J., Szekely, P.: MAAN: A Multi-Attribute Addressable Network for Grid Information Services. Journal of Grid Computing, vol. 2 n. 1 (2004) 3-14.
10. Oppenheimer, D., Albrecht, J., Patterson, D., Vahdat, A.: Scalable Wide-Area Resource Discovery. UC Berkeley Technical Report, UCB/CSD-04-1334 (2004).
11. Spence, D., Harris, T.: XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform. Proc. of HPDC'03, Washington, USA (2003).
12. Basu, S., Banerjee, S., Sharma, P., Lee, S.: NodeWiz: Peer-to-peer Resource Discovery for Grids. Proc. of IEEE/ACM GP2PC'05, Cardiff, UK (2005).
13. Fisk, A. A.: Gnutella Dynamic Query Protocol v0.1. http://www.thegdf.org/wiki/index.php?title=Dynamic_Querying.
14. El-Ansary, S., Alima, L., Brand, P., Haridi, S.: Efficient Broadcast in Structured P2P Networks. Proc. of IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGRID'05), Cardiff, UK (2005).
15. Andreozzi, S., Burke, S., Field, L., Fisher, S., Konya, B., Mambelli, M., Schopf, J., Viljoen, M., Wilson, A.: GLUE Schema Specification Version 1.2: Final Specification - 3 Dec 05. http://infnforge.cnaf.infn.it/glueinfomodel/index.php/Spec/V12.
16. Comito, C., Talia, D., Trunfio, P.: Grid Services: Principles, Implementations and Use. International Journal of Web and Grid Services, vol. 1 n. 1 (2005) 48-68.