Peer-to-Peer Resource Discovery in Grids: Models and Systems ⋆

P. Trunfio [a], D. Talia [a], C. Papadakis [b], P. Fragopoulou [b,∗], M. Mordacchini [c], M. Pennanen [d], K. Popov [e], V. Vlassov [f], S. Haridi [f]

[a] *DEIS, University of Calabria, Via Pietro Bucci 41C, 87036 Rende (CS), Italy*

[b] *Institute of Computer Science, Foundation for Research and Technology-Hellas P.O. Box 1385, 71 110 Heraklion-Crete, Greece*

[c] *Universita Ca Foscari di Venezia and INFN Sezione di Padova, Italy*

[d] *VTT Information Technology Tietoverkot, P.O.Box 1203 FIN-02044, Finland*

[e] *Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden*

[f] *School of Information and Communication Technology, Royal Institute of Technology, Electrum 229, SE-164 40 Kista, Sweden*

Abstract

Resource location or discovery is a key issue for Grid systems in which applications are composed of hardware and software resources that need to be located. Classical approaches to Grid resource location are either centralized or hierarchical and will prove inefficient as the scale of Grid systems rapidly increases. On the other hand, the Peer-to-Peer (P2P) paradigm emerged as a successful model that achieves scalability in distributed systems. One possibility would be to borrow existing methods from the P2P paradigm and to adopt them to Grid systems taking into consideration the existing differences. Several such attempts have been made during the last couple of years. This paper aims to serve as a review of the most promising Grid systems that use P2P techniques to facilitate resource discovery in order to perform a qualitative comparison of the existing approaches and to draw conclusions about their advantages and weaknesses. Future research directions are also discussed.

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

∗ Corresponding author.

Email addresses: `trunfio@deis.unical.it` (P. Trunfio), `talia@deis.unical.it` (D. Talia), `adanar@ics.forth.gr` (C. Papadakis), `fragopou@ics.forth.gr` (P. Fragopoulou), `mordacchini@dsi.unive.it` (M. Mordacchini), `mika.pennanen@vtt.fi` (M. Pennanen), `kost@sics.se` (K. Popov), `vlad@imit.kth.se` (V. Vlassov), `seif@imit.kth.se` (S. Haridi).

1 Introduction

The ultimate target of any resource sharing environment is to pool together large sets of resources and to make them available to its users and the deployed applications. A fundamental service in these environments is resource location which allows the discovery of resources across multiple administrative domains based on a list of predefined attributes. After specifying the attributes of existing resources, the system returns a list of locations where the required resources currently reside.

Grid and P2P are the two most common types of resource sharing systems currently in wide use. These two systems evolved from different communities and serve different needs. Grid systems interconnect computer clusters, storage systems, instruments, and in general available infrastructure of large scientific computing centers in order to make possible the sharing of existing resources, such as CPU time, storage, equipment, data, software applications. Most Grid systems are of moderate-size, they are centrally or hierarchically administered and there are strict rules governing the availability of the participating resources (i.e., a large percentage of the CPU time of a participating cluster should be dedicated for Grid use 24 hours a day). Grids are used for complex scientific applications which are time critical and they comply to strict QoS rules. Grid resources are highly dynamic and their values vary significantly over time (i.e., available CPU time, memory, available storage, network bandwidth). The resources required by applications are described by specifying a set of attributes (multi-attribute queries) like available computing power and memory. On the other hand, the most popular service provided by P2P systems is file sharing (e.g., Gnutella, KaZaA). Other applications are real time data transfer (e.g., telephony such as Skype), cycle stealing (e.g., SETI@Home), or collaboration (e.g., Groove). The typical participant in such systems enters from a common desktop asking to download music or video files over Internet TCP connections. Participation is highly dynamic as users can enter, depart, and rejoin the system totally unpredictably. Finally, most resource location queries are not attribute-dependent as in Grids but they either specify a file name, they are keyword searches or range queries.

Grid and P2P are both resource sharing systems having as their ultimate goal the harnessing of resources across multiple administrative domains. They have many common characteristics such as dynamic behavior and heterogeneity of the involved components. Apart from their similarities, Grid and P2P systems exhibit essential differences reflected mostly by the behavior of the involved users, the dynamic nature of Grid resources (i.e., CPU load, available memory, network bandwidth, software versions) as opposed to pure file sharing which is by far the most common service in P2P systems. Another essential difference results from the demanding nature of sensitive Grid applications that are time and data critical and have strict fault tolerance and security requirements as opposed to P2P applications which use commodity hardware and exhibit best effort behavior. The basic differences between Grid and P2P systems are summarized in Table 1.

Table 1
The basic differences between Grid and P2P systems.

In terms of ...	**Grids**	**P2P**
Users	Scientific community	Any desktop user
Computing	Workstations, multiprocessors, computer clusters	Common desktops
Network	High speed dedicated network	Internet TCP connections
Administration	Centralized or hierarchical	Distributed
Applications	Complex scientific applications such as large scale simulations, data analysis	File sharing, real-time data streaming, cycle steeling
Scale	Connect a relatively small number of specialized sites (moderate size)	Connection is possible from any desktop (large size)
Security	Secure services for job submission and interactive execution	Protocols for file sharing
Participation	Static or slowly changing participation of nodes over time	Nodes can enter or leave totally unpredictably
Trust	Trusted users	Untrusted, anonymous users

Although Grid and P2P systems emerged from different communities in order to serve different needs and to provide different functionalities, they both constitute successful resource sharing paradigms. It has been argued in the literature that Grid and P2P systems will eventually converge [50,24]. The techniques used in each of these two different types of systems will result to a mutual benefit. As the scale of Grid systems rapidly increases, centralized management will prove inefficient and other methods will be considered. The QoS constraints that currently govern most Grid applications will loosen up as Grids will move towards more popular and diverse application scenarios. Strict resource participation rules will be relaxed as participating organizations may need to have their infrastructure for own use at certain periods and for Grid jobs at other times and the use of commodity hardware will be allowed. On the other hand, P2P systems will open up to more sophisticated applications and they will have to support more complex queries and different QoS levels. P2P is not just for file sharing, but opens a new era of group based resource-aware communications for many new applications and services for connected lifestyles [25]. Adding together location information, presence information, and many other parameters available in the end-to-end communication path and local resources, new P2P service scenarios can be envisioned. The market is driving P2P services based on IP into wireless mobile devices. Some current examples of such services are VoIP, Instant Messaging, file sharing, navigation systems and multi-party video conferencing. Associated P2P/Grid application such as mobile Grid for access to Grid resources or distribute/store shared information across Virtual Organizations (VOs), such as web, digital cameras, own digital community-TV broadcast, music, etc, will also emerge. Tomorrow's resource sharing systems will be of large scale, highly dynamic (in terms of resource availability and infrastructure participation), and will exhibit high heterogeneity. Resources can become unavailable at any time and machines can enter or leave the resource sharing system unpredictably while QoS will still be an issue for specific applications.

The purpose of the present report is to serve as a review of the most promising Grid systems that incorporate P2P resource location methods, in order to perform a qualitative comparison of the existing approaches and to draw conclusions about their advantages and their weaknesses. The systems discussed in this report make use of different P2P approaches, ranging from unstructured [23,51,33,32,39] to structured ones [8,6,36,47]. Along with the presentation of the systems, their capabilities in terms of scalability, reliability, efficiency, ease of implementation, ease of use, self-organization, fault-tolerance, security, robustness are also discussed. Furthermore, we elaborate on P2P-based approaches for building scalable Grid resource discovery services based on semantic information in order to improve the precision of resource discovery [26,30,22,44,57]. Finally, future research directions in the field are discussed.

The remainder of this report is organized as follows. Section 2 discusses the P2P paradigm, describing the different models (including unstructured, structured, and hybrid) and comparing them on the basis of their performances and capabilities. Section 3 presents a review of systems that adopt a P2P approach to Grid resource discovery. Section 4 discusses the role of semantics in the design of P2P-based Grid resource discovery systems. Finally, Section 5 provides a comparison of the different P2P models and systems for resource discovery in Grid environments.

2 Resource discovery in Peer-to-Peer systems

The P2P paradigm is based on the principle that every component of the system has the same responsibilities acting simultaneously as a client and as a server, as opposed to the traditional client-server model. P2P systems are divided into two main categories based on the connection protocol they employ and the way peers are organized *structured* and *unstructured*. Structured P2P systems employ a rigid structure to interconnect the peers and to organize the file indices, while in unstructured systems each peer is randomly connected to a fixed number of other peers and there is no information about the location of files. Several hybrid approaches have been proposed to overcome the drawbacks of the two main approaches while retaining their benefits.

Napster was introduced in 1999 and is historically the first P2P system. It comprised a central server which stored the index of all files shared by the participating peers. To locate a file a user queried the central server using the name of the file and received as a result the IP address of a peer containing the file. A direct connection was established between the requesting peer and the peer containing the file in order for the download to be effected. The central index server used in Napster is not easy to scale and was a single point of failure. Although Napster is historically considered as the first unstructured P2P system, the existence of a central index differentiates it considerably from today's unstructured P2P systems.

In today's unstructured P2P systems, each peer maintains a constant number of connections to other peers, called its neighbors, thus an overlay network of peers is formed. Gnutella and KaZaA are considered two of the most popular unstructured systems (Fig. 1). Due to the lack of an underline structure in those systems, there is no information about the location of files, thus the prevailing resource location method is “flooding”. A peer looking for a file issues a query which is broadcast in the network. All matching query responses are send to the originating peer through the reverse path. Clearly flooding is not scalable since it creates a large volume of unnecessary traffic in the network. To limit the number of messages generated by flooding, each message is tagged with a *Time-To-Live* (*TTL*) field. The TTL indicates the number of hops away from its source a query should propagate. A small TTL value can reduce the network coverage, defined as the percentage of network nodes that receive a query, thus a file may fail to be located although present in the system.

In order to limit the vast amount of messages produced by flooding, modern unstructured P2P systems employ a controlled flooding mechanism, also known as *Dynamic Querying* [2]. In dynamic querying the peer that initiates a query tries to control the query's propagation by sending it only to a subset of its neighbors and with a small TTL. If this first attempt does not produce a sufficient number of results, the originating peer may broadcast the query again to a different set of neighbors with increased TTL. This process is repeated until a satisfactory amount of results is received, or until all the neighbors are exhausted. Many other techniques have been proposed in the literature to alleviate the excessive traffic problem caused by flooding and to deal with the traffic/coverage trade-off [52] such as *random walks*, *multiple random walks*, hybrid methods that combine flooding and random walks [18], directed searches based on statistical information [31], *forwarding indices* [12], or the incorporation of semantic information [13,48,55].

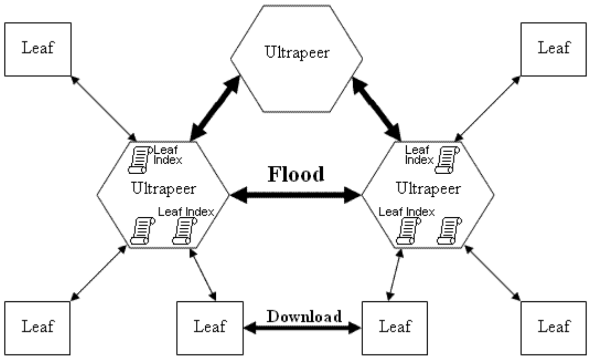

Fig. 1. The architecture of Gnutella.

Experiments demonstrated that peers with low bandwidth connections (i.e., nodes connected over dial-up modems) are easily saturated by flooding request and thus slow down resource location in unstructured P2P systems. To exploiting peer heterogeneity to the system's benefit low bandwidth peers had to be isolated from query routing. By curbing the definition of P2P systems, in [54,10] a distinction between peers was introduced and a two level hierarchy of peers was constructed. High bandwidth peers, the *Superpeers* (also known as *Ultrapeers*), formed an unstructured overlay network, while peers with low bandwidth, the *leaves*, were connected only to Superpeers. Each Superpeer has an index of all the files contained in its leaves. Any request originating at a leaf peer is forwarded through the Superpeers it is connected to, while flooding is performed only at the Superpeer overlay network. This modification allows the system to retain its simplicity while offering improved scalability.

2.2 Structured P2P systems

Structured P2P systems are equipped with a distributed indexing service which is based on hashing, and is known as *Distributed Hash Table* (*DHT*). Peers and files are mapped, usually through the same hash function, to a key space. Peers and file indices are organized in a rigid structure according to their keys, which facilitates the location of files. Most structured P2P systems support naturally exact match queries in $O(\log N)$ hops, where N is the size of the key space, and range queries. However they do not support directly keyword searches which constitute the core of queries in real P2P systems.

Chord [49] is the first structured P2P system to be proposed. In Chord, both peers and files are mapped through the same hash function to an m-bit key space. The peers in Chord are organized in an 1-dimensional circle according to their keys 2. Each peer stores the index of all files whose keys fall in the range between the key of its predecessor and its own key. The lookup process emulates the binary search, thus requires $O(\log N)$ steps and messages. The

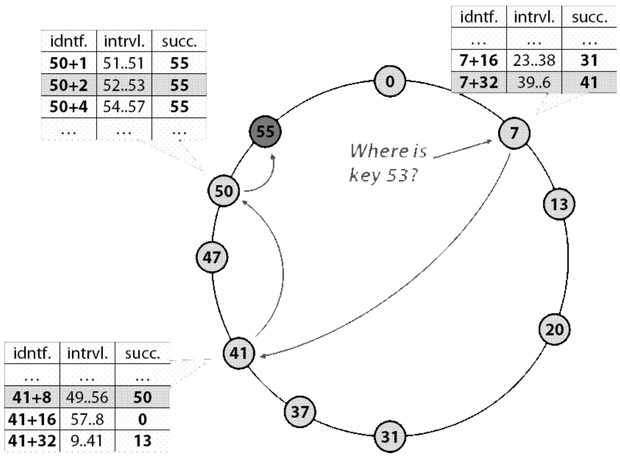

Fig. 2. The architecture of Chord.

arrival or departure of a peer does not have a global effect but affects at most $\log N$ other peers. Whenever a peer joins the network, it takes responsibility of certain file keys previously assigned to its successor. When a peer leaves the network, all of its assigned file keys become the responsibility of its successor. In general, each peer is responsible for an equal number of keys with high probability, thus load balancing is achieved.

The *Content-Addressable Network* (*CAN*) [40] tries to limit to a constant the number of each peer's neighbors, regardless the size of the network or the key space. The peers in CAN are organized in a d-dimensional torus. Each peer's key consists of d numbers and corresponds to a point in a d-dimensional space. Each peer is connected to its next and previous peer in each dimension, thus having $O(d)$ neighbors. The d-dimensional space is divided equally among the available peers and each peer is responsible for all file keys corresponding to points in its own subspace. Each lookup request can be forwarded over any of the d dimensions, leading to a lookup cost of $O(N^{1/d})$ in time and number of messages. Peer arrivals and departures in CAN have a very localized effect, since they only affect $O(d)$ other peers. In order for a new peer to join, it contacts a peer already in the system, which splits its subspace in two halves and assigns responsibility of one halve to the new peer. Neighbors are set accordingly. An extension of CAN employs more than one hash functions in order to support replication and thus to reduce lookup cost and to provide fault tolerance in case of unpredictable peer departures.

Koorde [16] has the same lookup costs as Chord, while maintaining a constant number of neighbors per peer. This is achieved by exploiting the properties of the de Bruijn graph. Each peer in Koorde is mapped to a binary key and is connected to two other peers, whose keys are formed by shifting the peer's key once to the left, dropping the high order bit and inserting as low order bits 0 and 1. File lookup is performed by emulating routing in the de Bruijn graph and thus requires $O(\log N)$ hops.

Recent examples of structured systems include Tapestry [56], Cyclone [43], and HiPeer [53]. Tapestry is very similar to Pastry but it includes a mechanism for hotspot alleviation. Although uniform hashing assigns ID keys to data uniformly, the high replication of some files, or their popularity may lead to load imbalances, something that Tapestry tries to avoid. Cyclone and HiPeer are examples of hierarchical DHTs. Hierarchy is added to the common, flat DHT design for reasons such as fault isolation, bandwidth utilization, adaption to the underlying network, etc. Cyclone forms subnetworks inside a DHT by dividing an ID key into a group ID part and a node ID part. However, in contrast to common practice, it assigns the lowest priority ID bits to the group ID. As a result, the nodes of a single group are distributed over the entire ID space. HiPeer, just like Koorde, exploits the attributes of De-Bruijn graphs. Unlike Koorde however, it creates a structure of co-centric rings, each being a De-Bruijn graph. Each outer ring can contain double the number of peers of the immediate inner ring. Peers in the inner rings are assumed to be more reliable than peers in the outer rings. The rings are interconnected in a way that De-Bruijn routing can be used to move between rings the same way it is done for peers of the same ring. This capability of "horizontal" as well as "vertical" routing enables HiPeer to locate a file in a number of hops equal to the number of rings in the system.

Structured P2P systems are more scalable than unstructured ones, in terms of traffic load, but need to have strong self-organization capabilities in order to be able to maintain their rigid structure. Structured systems are prone to node failure, and unpredictable node departures. Although in the past few years considerable effort has been devoted the research on structured P2P systems, they have also earned a lot of criticism for their high maintenance cost in the presence of high churn, their difficulties to support more general queries, and their exclusive support for exact matches which constitute a relatively small percentage of queries in real P2P systems [10].

2.3 Hybrid approaches

Both unstructured and structured approaches have advantages and disadvantages. Several hybrid approaches have been proposed to overcome the drawbacks of each while retaining their benefits.

In *Pastry* [42] each peer is mapped to a random 128-bit node identifier (nodeId). Pastry nodes are organized in a circle according to their nodeIds. Each peer recursively divides in two parts the space its nodeId belongs. The target is for each peer to maintain knowledge of at least one peer belonging to each resulting subdivision. Each lookup message is routed to the peer whose nodeId has the longest common prefix with the lookup Id. After each routing step, the common prefix of the lookup Id with the current peer Id increases in length by one. Thus, each file can be located in $\log N$ steps. For robustness, peers in Pastry maintain knowledge of the $2k$ closest peers, instead of just the next and the previous ones.

In *Kademlia* [34] each peer is mapped to a 160-bit key through the SHA-1 hash function. Each peer subdivides the space of possible distances between any keys, defined as their XOR. Each peer is aware of at least one peer, whose distance from its key is between $2i$ and $2i+1$, for $0 \leq i < \log N$. Those ranges are called "buckets". For redundancy and fault-tolerance, each peer tries to maintain knowledge of k peers in each bucket. For the same reason, each file key is also stored on the k peers closest to its key. Routing is performed by calculating the XOR of the requesting peer's key with the lookup key and forwarding the lookup request to a peer of the appropriate bucket. Kademlia peers monitor incoming traffic to become aware of alive peers in the network in order to update their buckets with more "fresh" contacts, at no cost. Resorting to lookups to refresh a bucket's contact is thus performed rarely, usually by new peers, during their bootstrap phase. Furthermore, there is no need for a departing peer to leave gracefully, since stale bucket entries are purged.

Kademlia and Pastry are similarly structured. Their structure exhibits less "strictness" compared to Chord and CAN, in the sense that for each defined subspace, any peer belonging to that subspace can serve as a contact. In Chord and CAN, all neighbor connections are strictly defined. Kademlia is the first hybrid P2P system to achieve global-scale deployment.

2.4 Comparison summary

The global-scale deployment of P2P systems makes scalability a very important issue. Unstructured P2P systems, lack in this aspect, due to the traffic generated by flooding. Improvements such as random walks have been proposed, to reduce the traffic generated, at the great expense of increased response time and reduced network coverage. Other proposals include sending the broadcast only to neighbors that have the highest history of returning results, or even disconnect from neighbors that do not return enough results (neighbor selection). On the other hand, structure makes scalability feasible, but it is difficult to maintain under high churn. This turns out to be a serious consideration since P2P systems are intended for the intermittent user that joins, departs, and rejoin the system totally unpredictably. Furthermore, for each data item in the system, the peer with the appropriate Id must be notified periodically. This results to either increased traffic, if the period is too small, or stale information (a peer holds information for a file shared by another peer that has left the system), if the period is too large. Although this problem is not vital in file-sharing P2P systems, it would be an issue if structured P2P systems are going to be used in Grid resource discovery, since resources are highly dynamic (CPU load, free memory, etc). In unstructured systems, where each peer answers queries about its own data only, each request will be checked against the latest data. On the other hand, the order and the data locality inherited in structured P2P systems is a valuable asset for Grid resource discovery, since it enables ranged queries to be performed efficiently. In contrast, to perform range queries in unstructured systems, one would have to contact every single peer.

In Table 2 a qualitative comparison between unstructured and structured systems is attempted based on the following criteria:

- *Scalability (time)*: Number of hops a query propagates;
- *Scalability (traffic)*: Number of query messages required;
- *Robustness*: Resilience under high churn;
- *Periodic update*: Need for data to be republished periodically;
- *Range queries*: Ability to efficiently support range queries.

Table 2
Unstructured vs structured P2P systems.

P2P system	**Scalability (time)**	**Scalability (traffic)**	**Robustness**	**Periodic update**	**Range queries**
Unstructured	$O(\log N)$	N*avg degree	High	No	No
Structured	$O(\log N)$	$O(\log N)$	Lower	Yes	Yes

Regarding system scalability in time and traffic the worse-case bound is given for a system with N peers. Furthermore, we assume unstructured P2P system of N nodes with a random overlay, thus $O(\log N)$ diameter.

3 P2P-based Grid resource discovery systems

As the Grid size increases, centralized and hierarchical approaches to Grid information systems do not guarantee scalability and fault tolerance. As pointed out earlier, a practical approach towards scalable solutions is offered by P2P models. The remainder of this section reviews some recently proposed systems that adopt a P2P approach to Grid resource discovery. As in the previous sections, also here frameworks are classified into unstructured and structured systems.

Iamnitchi et al.

In [23] Iamnitchi et al. propose a fully decentralized P2P architecture for resource discovery in Grid environments. In this architecture every participant in a Virtual Organization (VO) publishes information on one or more local servers, called *nodes* or *peers*, that store and provide access to local resource information. A node may provide information about one resource (e.g., itself) or multiple resources (e.g., all resources shared by an organization). Users send their requests to some known (typically local) node. The node responds with a matching resource description if it has them locally, otherwise it forwards the requests to another node. Intermediate nodes forward a request until its TTL expires or matching resources are found, whichever occur first. If a node has information matching a forwarded request, it sends the response directly to the node that initiated the forwarding, which in turn will send it to its user.

The architecture partitions the resource discovery solution into four components: *membership protocol*, *overlay construction*, *preprocessing*, and *request processing*. The membership protocol specifies how new nodes join the network and how nodes learn about each others. The overlay construction function selects the set of collaborators from the local membership list. Preprocessing refers to off-line processing used to enhance search performance prior to executing requests. The request processing implements the request propagation strategy. This strategy decides to which node (among the locally known ones) a request is to be forwarded. In addition to contact addresses, nodes can store additional information about their neighbors, such as statistical information about previously answered requests. The tradeoff between the amount of information kept for each neighbor and the search performance generates four request propagation strategies: *random walk*, *learning-based*, *best-neighbor*, *learning-based + best-neighbor*.

In the random walk strategy the node to which a request is forwarded is chosen randomly. In the learning-based strategy nodes learn from experience by recording the requests answered by other nodes. A request is forwarded to the peer that answered similar requests previously. If no relevant experience exists, the request is forwarded to a randomly chosen node. The best neighbor algorithm records the number of answers received from each peer, and a request is forwarded to the peer that answered the largest number of requests. Finally, the learning-based + best-neighbor strategy is identical with the learning-based strategy except that, when no relevant experience exists, the request is forwarded to the best neighbor.

Experimental results obtained on a Grid emulator showed that the learning-

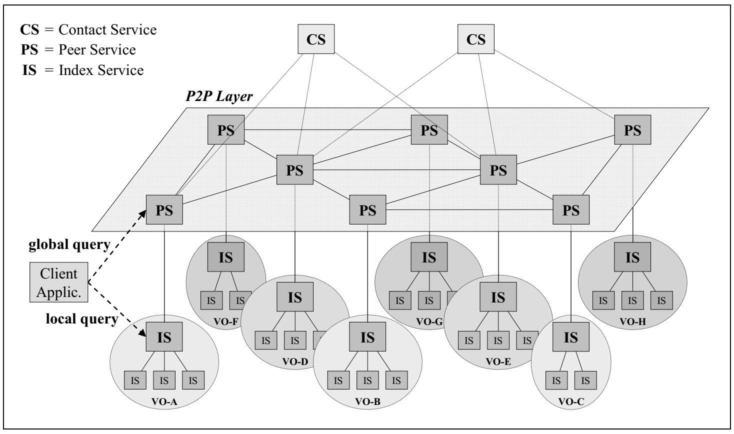

Fig. 3. The architecture proposed by Talia et al. [51].

based strategy is the best regardless of request distribution. Key to the performance of the learning-based strategy is the fact that it takes advantage of similarity in requests by using a possibly large cache. It starts with low performance until it builds its cache. As expected, the random-forwarding algorithm resulted the least efficient, but has the advantage that no additional storage space is required on nodes to record history.

Talia et al.

In [51] Talia et al. propose a P2P architecture for resource discovery in OGSA-compliant Grids. The architecture is composed of two layers (see Fig. 3): the lower one is a hierarchy of *Index Services* (as provided by Globus Toolkit versions 3 and 4), which publish information owned by each VO; the upper one is a *P2P Layer*, which collects and distributes this information. The P2P Layer includes two types of OGSA-compliant Web Services: *Peer Services* used to perform resource discovery, and *Contact Services* that allow Peer Services to organize themselves in a P2P network.

There is one Peer Service per VO. Each Peer Service is connected with a set of Peer Services - its neighbors - and exchanges query/response messages with them in a P2P mode. A connection between two neighbors is a logical state that enables them to directly exchange messages. Direct communication is allowed only between neighbors. Therefore, a query message is sent by a Peer Service only to its neighbors, which in turn will forward it to their neighbors. A query message is processed by a Peer Service by invoking the top-level Index Service of the corresponding VO. A query response is sent back along the same path that carried the incoming query message. To join the P2P network, a Peer Service must know the URL of at least one Peer Services to connect to. An appropriate number of Contact Services is distributed in the Grid to support this procedure. Contact Services cache the URLs of known Peer Services; a Peer Service may contact one or more well known Contact Services to obtain the URLs of registered Peer Services.

An extension of the Gnutella protocol is adopted to exchange discovery messages among Peer Services at the P2P Layer. This protocol uses ad hoc techniques to make Web Services effective as a way to exchange discovery messages in a P2P fashion. In particular, two main strategies are adopted (1) *message buffering*: messages to be delivered to the same peer are buffered and sent in a single packet at regular time intervals; and (2) *message merging*: messages with the same header (i.e., same type, identifier, and receiver) are merged into a single message with a cumulative body. Experimental results showed that appropriate message buffering and merging strategies produce significant performance improvements, both in terms of number and distribution of Web Service operations processed.

Mastroianni et al.

In [33] Mastroianni et al. adopt the *super-peer* model to design a P2P-based Grid information service. The super-peer model has been originally proposed to achieve a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and fault-tolerant features offered by distributed search [54]. A super-peer node acts as a centralized server for a number of regular peers, while super-peers connect to each other to form an overlay network that exploits P2P mechanisms at a higher level.

The super-peer model is advantageously exploited in the Grid context because it is naturally appropriate for large-scale Grid environments. In fact, a large-scale Grid can be viewed as a network interconnecting small-scale, proprietary *Physical Organizations* (*PO*s), where each PO is composed of a set of Grid nodes within one administrative domain. Within each PO, one or more nodes (e.g., those with the largest capacity) act as super-peers, while the other nodes use super-peers to access the Grid and search for resources and services. A super-peer has two major roles: it is responsible for the communication with the other POs and it maintains metadata of all nodes in the local PO.

The resource discovery protocol works as follows. Query messages generated by a Grid node are forwarded to the local super-peer. The super-peer examines the local information service to verify if the requested resources are present in the nodes of the local PO. If this is the case it sends to the requesting node a queryHit containing the IDs of the nodes containing the requested resources. Otherwise, the super-peer forwards a copy of the query to a selected number of neighbor super-peers, which in turn contact the respective information system, and so on. Whenever a resource matching the query criteria is found in a remote PO, a queryHit is generated and is forwarded along the same path to the requesting node, and a notification message is sent by the remote super-peer to the node that handles the discovered resource. The set of neighbors to which a query is forwarded is determined based on statistical information about previous queryHits received from the the neighboring super-peers. Moreover, a number of strategies are adopted to decrease the network load, reduce the response time, and increase the probability of success (i.e., the probability that a query issued by a peer will be followed by at least one queryHit).

Puppin et al.

Puppin et al. propose another Grid information service based on the super-peer model [39]. Grid nodes are grouped into clusters, where each cluster may include one or more super-peer nodes. The system defines two main components: the *Agent* and the *Aggregator*. The Agents works as an OGSA-compliant Grid Service available at each network node. It publishes all information made available by the *information providers*. The information providers periodically query the resources and store the gathered information as *Service Data Element* (*SDE*). When a resource is published, the name of its Service Data is broadcast to all the Aggregators in the cluster.

Aggregators work as super-peers, acting as servers within their cluster, and as peers in the network created by all the Aggregators. Each Aggregator is thus responsible for collecting data, replying to queries, forwarding queries to other Aggregators, and keeping an index of the information stored in each neighbor Aggregator. The *Hop-Count Routing Index* (*HRI*) is used in this system to improve the performance of routing and to prevent the P2P network from being flooded. The HRI is used to exchange queries among super-peers and, in particular, to select the neighbor super-peers with the highest probability of success.

Marzolla et al.

In [32] Marzolla et al. propose another system for discovering Grid resources based on routing indexes. In this system, nodes are organized in a tree-structured overlay network, where each node maintains information about the set of resources it manages directly and a condensed description of the resources present in the sub-trees rooted in each of its neighboring nodes. The data location algorithm exploits those indexes to route queries towards areas where matches can be found.

Data about resources is mapped in the following way. For each possible attribute of a resource item, the domain of the attribute is split into k sub-intervals. The value of k may differ from one attribute type to another. Given a resource, the index for attribute A of the resource is represented by a k-bit vector, with all of its entries set to 0, except the one corresponding to the sub-interval that contains the actual value of A. The index that represents attribute A for all the local resources of a peer P is easily obtained by performing a logical `OR` operation of all the attribute bit vectors of the local items. Moreover, for each attribute, node P receives from each neighbor N_p an index that is the bitwise union (`OR`) of the local bit vectors of all peers present in the sub-tree routed at N_p.

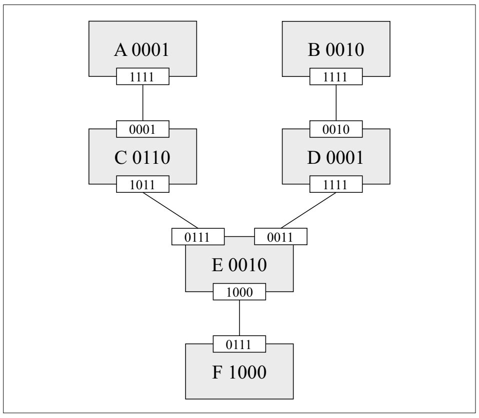

Fig. 4. Example of a network with bit vector indexes, as proposed by Marzolla et al. [32].

When a peer P receives a multi-attribute range query it decomposes it into a set of sub-queries, one per attribute. Each sub-query Q is then mapped into a binary vector of the same length k of the vector that represents the attribute related to the query. All the entries that correspond to sub-intervals contained in the range specified by the sub-query are set to 1. Subsequently, the sub-queries are first matched against local indexes, in order to find out whether there are local resources satisfying the query. Then, the sub-queries are matched against the routing indexes and are eventually routed only to those neighbors whose indexes satisfy all the sub-queries. Whenever there are data value changes, update messages are sent to neighbors only if the new bitmap representation of the resources differs from the old one, already known by the neighbors. Simulation results show that the proposed update and query processing algorithms have good scalability properties, meaning that query messages are routed to a relatively small number of peers without flooding the network and update messages involve a constant number of peers, regardless of network size.

3.1.1 Comparison Summary

The unstructured systems described in this section adopt different architectures, including flat P2P networks [23,51], tree-based overlays [32], and super-peer networks [33,39]. In particular, some experiments [33] show that the super-peer model is naturally appropriate to the organization-based nature of current Grids, ensuring limited network load and reduced response time with respect to pure-decentralized P2P systems. Moreover, diverse strategies have been adopted to provide up-to-date results with limited network traffic, including experience-based query forwarding [23,33], message buffering and merging [51], and routing indexes [39,32]. All this strategies experimented significant benefits with respect to naive techniques, such as simple flooding. Table 3 summarizes the main features of the unstructured systems described above in terms of architecture, resource indexing, and query resolution.

Table 3
Qualitative comparison of Grid discovery systems based on unstructured P2P architectures.

System	Architecture	Resource indexing	Query resolution
Iamnitchi et al. [23]	Flat P2P overlay network, including one or more peers per VO.	Each peer provides information about one or more resources.	Queries can be forwarded using different strategies: random walk, learning-based, best-neighbor, learning-based + best-neighbor.
Talia et al. [51]	Flat P2P overlay network, including one OGSA-compliant Peer Service per VO.	Within each VO, a hierarchy of Index Services provides information about local resources.	Discovery messages are routed across Peer Services using a modified Gnutella protocol. Message buffering and merging techniques are used to reduce Web Service overhead.
Mastroianni et al. [33]	Within each organization, one or more nodes act as super-peers.	A super-peer maintains metadata of all nodes in the local organization.	The set of super-peers to which a query is forwarded is determined on the basis of statistical information about previous discovery tasks.
Puppin et al. [39]	Nodes are grouped into clusters, where each cluster may include one or more super-peer nodes.	On each node, an Agent publishes information about resources. The information is broadcast to all super-peers in the cluster.	The Hop-Count Routing Index is used to select the neighbor super-peers with the highest probability of success.
Marzolla et al. [32]	Nodes are organized in a tree-structured overlay network.	Each node maintains a condensed description of the resources present in the sub-trees rooted in each of its neighboring nodes.	A multi-attribute query is decomposed into a set of sub-queries. The sub-queries are matched against the routing indexes and routed only to those neighbors whose indexes satisfy all the sub-queries.

3.2 Structured systems

MAAN

The authors of MAAN [8] propose an extension of the Chord protocol to handle multi-attribute range queries. Each node of the system is part of a Chord overlay network. The values of the resources are mapped to the Chord m-bit key space using a uniform locality preserving hash function and having one different registration for each of the resource attributes. Each registration is composed by a pair $<$ *attribute-value*, $resource\text{-}info >$. Each node is responsible of maintaining the information of the registered keys that fall into the key space sector it supervises.

The resolution of multi-attribute range queries is implemented in two different ways. The first one is an iterative approach: if a query is composed of M sub-queries, each sub-query is resolved separately in the proper attribute space. The results are then collected and intersected at the query originator node. This is the most simple and, at the same time, inefficient way of resolving queries. Its complexity is $O(\sum_{i=1}^{M}(logN + N \times s_i))$, where M is the number of sub-queries, N the number of peers and s_i the selectivity of sub-query i. The second method is defined as a single attribute dominated routing. Let X be the set of resources that satisfies query Q. Then X should satisfy all the sub-queries of Q, so we have $X = \bigcap_{1 \leq i \leq M} X_i$, where X_i is the set that satisfies the sub-query on attribute a_i. The system uses the Chord to find a single set of candidate resources X_k for attribute a_k. X_k is a superset of X, so all the solutions for query Q are contained in X_k. Since all the resources store a $<$ *attribute-value*, $resource\text{-}info >$ pair, it is possible to exploit the resource-info field to find the X_k's resources that match all the other sub-queries. This method has a complexity of $O(logN + N \times S_{min})$, where S_{min} is the minimum selectivity for all attributes. Load balancing of resources is achieved by constructing a locality preserving hash function which produces a uniform distribution of hash values. In order for the construction of this hash function to be feasible the distribution of input attribute values should be continued, monotonically increasing, and known in advance, which is the case for many common distribution functions.

Andrzejak et al.

In [6] Andrzejak and Xu propose an extension of the DHT-based CAN system to allow range queries for a Grid information service. In this framework, all Grid resources are described by a set of attributes. For each attribute either a standard DHT or the proposed CAN extension is used depending on its type. In particular, attributes which have a limited number of values are handled by standard DHT systems, while for "continuous" types of attributes the extended CAN system is adopted. To locate resources specified by several attributes, the information infrastructure queries for each attribute present in the query the appropriate DHT and then concatenates the results in a database-like "join" operation.

A subset of the servers participating in the Grid acts as nodes in a CAN-based P2P-network and store the pairs $<$ *attribute-value*, $resource\text{-}ID >$. Each one of them is responsible for a certain subinterval of the attribute values. Such a server is called an *Interval Keeper* (*IK*) and the corresponding subinterval its *interval*. Each server in the Grid reports its current attribute value to an IK with the appropriate interval. The authors propose different strategies for propagating range-query requests and to minimize the communication overhead during the attribute updates. The effectiveness of these strategies have been demonstrated through simulations using both synthetic and real-life workloads.

SWORD

SWORD [36] locates a set of machines matching user-specified constraints on both static and dynamic node characteristics, including both single-node and inter-node characteristics. SWORD provides a range of mechanisms and functionalities, including: (1) techniques for efficient handling of multi-attribute range queries that describe application resource requirements; (2) an integrated mechanism for scalably measuring and querying inter-node attributes without requiring $O(n^2)$ time and space; (3) a mechanism for users to encode a restricted form of utility function indicating how the system should filter candidate nodes when more are available than the user needs; and (4) an optimizer that performs this node selection based on per-node and inter-node characteristics.

In SWORD there are two kinds on nodes: *reporting nodes*, which periodically send measurement reports on resources, and *DHT server nodes*, which receive resource information and handle queries from users. Server nodes are organized using Bamboo, which is a structured peer-to-peer system similar to Pastry. For each of the n single-node attributes $A_1, A_2, \ldots, A_n$ that can appear in a query, each reporting node periodically sends a tuple of all its attribute values to n DHT keys $k_1, k_2, \ldots, k_n$, where each k_m is computed based on the corresponding value of attribute A_m. Upon receiving such a tuple, a server stores the tuple in a hash table indexed by the identity of the node described by the report. For each attribute A, the range of possible values is mapped to a contiguous region of the DHT keyspace using a given function f_A. Thus, a list can be obtained of all nodes that are reporting A values in some range $x_{min} - x_{max}$ by visiting the DHT nodes that "own" all DHT keys between $f_A(x_{min})$ and $f_A(x_{max})$.

To issue a query, a user opens a TCP connection to any SWORD instance and sends the query. The contacted SWORD instance initiates the distributed range search, followed by the retrieval of the needed inter-node measurements and the invocation of the optimizer. That node then returns the result to the user over the same TCP connection.

XenoSearch

The system proposed in [47] exploits and extends the Pastry indexing and routing system. XenoSearch allows multi-dimensional searchings by constructing a separate Pastry ring for each resource attribute. A peer (XenoServer) registers itself separately in each ring. Range queries for a single attribute are possible thanks to the fact that the information is conceptually stored in a tree where the leaves are the XenoServer nodes. The tree internal nodes are called *Aggregation Points* (*AP*). Each AP summarizes the range of values of the nodes below it in the tree. An AP is distinguished by a key in the same key space as the attributes. The key of an AP is a prefix of the keys of its child nodes. By knowing the key of an AP, we can determine the range of values of the leaf-nodes of that AP, i.e., the value range of the leaf-nodes XenoServers attributes. An AP key is mapped into the Pastry ring and the closest XenoServer in the key space is in charge for maintaining the information related to that AP.

Multi-attribute queries are resolved by decomposing each query in a set of sub-queries, one per attribute. Then, for each sub-query, a single attribute range query is performed. The retrieved results are then intersected in order to find the final set of resources that satisfies the original query. The client that originated the query is given a set of possibly matching XenoServers. The client has to further query the nodes to know the real server's resource state. This is necessary because the information in the system is refreshed only periodically. Thus, the results obtained by XenoSearch may not be always up-to-date.

Mercury

Mercury [7] is a Grid system that supports multi-attribute queries. It uses Symphony, a one-dimensional DHT as its underlying architecture. Each single attribute is assigned to a different DHT, called the Hub. Each resource is registered to the hub of each different attribute in its attribute set. To avoid querying more than one Hub during the resolution of a multi-attribute query, each resource stores all its attribute-value pairs in all Hubs it is registered. Since each hub indexes the resources it stores according to only one attribute, a range query is resolved based on one attribute only. Thus, a multi-attribute query is resolved by selecting the attribute with the smallest range, and querying the appropriate hub. The query uses the underlying DHT system to locate the resources with the smallest value in the range of the query. It then proceeds to the next values, until the largest value in the range of the query. The query responds with a list of all the resources in the traversed range whose other attribute values also matched the corresponding ranges in the query.

Load-balancing is performed by periodically probing the system to find load-imbalances. A Symphony graph has been proven to be an expander, meaning that a random walk of $logN$ hops is enough to perform a near-perfect uniform random sampling of the network. Using this random walk, a heavily loaded node can locate a lightly-loaded one. Upon this discovery, the first node will send a special message to the second node, which will leave the network and rejoin in such a way that it will become neighbor of the first node, thus sharing its load (leave-rejoin protocol).

Schmidt et al.

The system proposed in Schmidt et al. [45] supports multi-attribute queries by using a single one-dimensional DHT. A space filling curve is used, to map all possible d-dimensional attribute values to a single dimension. In particular, each resource with a set of attribute values is mapped to the node whose ID is generated by interleaving the binary representations of its attribute values. For example a resource with three attributes with values (1 (01), 2 (10), 3(11)) will be stored in node with ID 011101. Notice here that if one needs to store resources with many attributes and each attribute has a wide range of possible values, this might require a DHT with IDs of many bits. However, the number of contacts a node has to maintain in a DHT of logarithmic lookup time, increases linearly with the number of bits. Each resource computes its node's ID according to its attribute values.

Range queries are similar to point queries but may contain some undefined bits. For example, a query of resources with attribute values (1, 2, 0-3) is 01*00*. Notice that range query sizes can only be powers of two and can only start from values that are also powers of two. For instance, we can not make a query of the form (1, 2, 1-3). Ranged queries are resolved much like point queries, albeit, when an undefined query bit is encountered the query will be propagated in more than one directions. The query is propagated to any node with additional common prefix bit with the query ID than the present node. Thus, if the querying node's ID is 1*****, it will forward the query to any node whose ID is in the form 0*****. In turn, that node will propagate it to any node of the form 01****. That node, in turn, will propagate the message twice. Once to a node of the form 01000* and once to a node of the form 01100* and so on. Tree-like structures usually suffer in the sense that lookup always starts at the root node, transforming it into a bottleneck. However, in this case, any node with a first ID bit same with the first ID bit of the query can be used as a root node, thus eliminating this drawback.

Ratnasamy et al.

Ratnasamy et al. [41] propose a system that utilizes a uniform hash function to distribute the storage load evenly across the participating nodes. Since the locality of attribute values is not preserved, another overlay on top of the underlying DHT is used to enable efficient range query resolution. All attributes are stores in a common DHT, however a different overlay structure is used for each attribute. Each resource registers itself in one overlay structure for every attribute it contains. A multi-attribute query is resolved in parallel in each overlay structure (separately for each attribute it contains) and the intersection of the results is calculated at the query originator node. The overlay structure used is a binary tree called a *tier*. There is a different tier for each attribute. The root node of a tier is assigned the whole attribute space while each one of its children is assigned half the range. Each resource is registered only at the leaf node whose range contains its attribute value. The recursive subdivision of a node's range occurs only when the node becomes overloaded with resources. Initially, only the root node exists and all resources register to it. When the number of resources becomes high, the root node creates two children nodes and splits the load to them. Each tier node is assigned to a DHT node at random using a uniform hash function. For instance, a tier node of attribute A responsible for the range of values from x to y will be mapped to the DHT node with ID = hash(A, x, y).

Lookup is performed by recursively dividing the attribute value range by two, to find the smallest range that contains the whole of the query range. Then, the DHT lookup functionality is used to find the node responsible for that range. Then, the subtree of that node is broadcasted, to locate all leaf nodes of that tree. This leads to a logarithmic time cost and 2*P*n message cost. Since hashing is used, storage load is not an issue. However, even though a query start by locating the root of the smallest subtree that contains the required range (and not the absolute root), the points where a range is subdivided is static. This means that, any query range, however small, that spans a border where two siblings are divided, will need to start from their parent.

3.2.1 Comparison Summary

Table 4 summarizes the main features of the structured systems described above in terms of architecture, protocol, resource registration, query resolution, and load balancing. As shown in the table, most architectures either adopt one DHT for all attributes [36,45], or arrange attribute values on multiple DHTs, one per attribute [8,6,47,7]. Both single-DHT and multi-DHT approaches have proved effective. Multi-DHT architectures are easier to implement and provide multi-attribute search capabilities in a simple way. On the other hand, in terms of required memory, multi-DHT architectures can be more expensive than single-DHT ones in case of resources with a high number of attributes.

All structured systems described in this section provide support for multi-attribute queries. To resolve multi-attribute queries, most systems resolve subqueries in parallel and intersect the results at the querying node [8,6,47,41]. Different solutions have been proposed to reduce the complexity of this task. For example, MAAN uses single attribute dominated routing, but its convenience depends on the selectivity of the attributes. Another approach is that

Table 4
Qualitative comparison of Grid discovery systems based on structured P2P architectures.

System	Architecture	Basic protocol	Multi-attribute resource registration	Multi-attribute query resolution	Range query resolution	Load balancing
MAAN [8]	One DHT per attribute	Chord	Each attribute is registered in the appropriate DHT	Each sub-query is resolved separately and the results are intersected at the querying node. Single attribute dominated routing	Sequential	Uniform, locality preserving hash function. Value distribution is known in advance
Andrzejak et al. [6]	One DHT per attribute	CAN	Each attribute is registered in the appropriate DHT	Each sub-query is resolved separately, and results are intersected at the querying node	Flooding	Simple neighbour load exchange
SWORD [36]	Each attribute is assigned a different subregion of a common DHT	Bamboo	Each attribute is registered in the appropriate region of the common DHT	The query is sent to the sub-region of the most selective attribute, or an attribute chosen at random	Tree-like	Leave-rejoin protocol. Customized hash functions
XenoSearch [47]	One DHT per attribute	Pastry	Each attribute is registered in the appropriate DHT	Each sub-query is resolved separately and the results are intersected at the querying node	Tree-like	None
Mercury [7]	One DHT per attribute	Symphony	All attributes are registered in every DHT	Lookup on the DHT of the attribute with the smallest range	Sequential	Periodical network sampling to find load-imbalances (leave-rejoin protocol)
Schmidt et al. [45]	One DHT for all attributes	Chord	Point query to register the attribute	Ranged query contains unknown bits. Each step forwards query to neighbour with an additional common prefix bit. Forward twice for each unknown bit	Tree-like	Exchange of load between neighbors
Ratnasamy et al. [41]	A range dividing tree per attribute. All trees mapped in a single DHT	Any	Each attribute is registered in the appropriate tier	Each sub-query is resolved separately and the results are intersected at the querying node	Tree-like	Uniform hash and attribute range subdivision

of Mercury, which adopts a sequential search starting from the attribute with the smallest range. Even in this case, the convenience of the approach in terms of response time may depend on the distribution of the value ranges.

Another important feature of most structured systems discussed here is the explicit management of load balancing [8,6,36,47,7,45,41], which is of main importance in large-scale Grids. The adopted solutions include: exchange of load between neighbours [6,45], leave-rejoin protocols [36,7], and the use of customized hashing functions [8,36,41]. The first two approaches are more effective but more expensive, since they require to periodically redistribute the workload by rearranging information. On the other hand, the use of customized hashing functions requires that the distribution of attribute values is known in advance, which could not be feasible in some contexts.

4 Grid resource discovery based on semantic information

As resource discovery in Grids is about finding relevant resources, the overall quality of a discovery service is determined not only by usual QoS measures such as performance, reliability and availability, but also by its precision that measures how many of the discovered resources are relevant, and how relevant they are. Precise resource discovery should be able to find *best approximate* matches usable for the requester. Resource discovery in Grids has to deal with a large number of volatile resources described using different approaches and languages, and managed by distinct VOs. In such heterogeneous and dynamic environments, syntactic keyword and taxonomy-based matching is insufficient to achieve high precision resource discovery. In order to improve the precision of a discovery service, resources must be given well-defined meaning carried by semantic information added to resource descriptions [15,19,17,9]. This semantic metadata describe the capabilities, interface, and internal organization, as well as the functional and non-functional properties of a resource. Semantic information is defined in terms of concepts and relations specified in an ontology. *Ontology* is an explicit specification of a conceptualization that serves as a foundation for formal representation of knowledge . It formally specifies how to represent objects, concepts and other entities that are assumed to exist in some area of interest and the relationships among them . In semantic-based resource-discovery, common ontologies facilitate communication between service providers and consumers.

Several researchers proposed to use (un)structured P2P networks as a medium for propagation of service discovery queries. For example, *Web Services P2P Discovery Service* (*WSPDS*), a fully decentralized and interoperable discovery service with semantic-level matching capability is presented in [26]. The discovery service is provided and consumed in a P2P network of WSPDS peers called *servents* where a servent can receive discovery requests from its user and its neighboring servents, and resolve the requests by querying its local site for matching services or/and by propagating requests to the neighboring servents. The authors present two architectures of the discovery service: (1) an unstructured P2P network of servents based on the Gnutella protocol and a keyword-matching where the servents collaborate to propagate discovery queries based on the probabilistic TTL-bounded flooding dissemination mechanism; (2) a semantic-enabled content-based P2P network of WSPDS servents called a *Querical Data Network* (*QDN*) [27] where identity for each node is defined by its data content. In the second prototype, Web Services Description Language (WSDL) service descriptions are augmented using DAML-S, which is a Web Service ontology based on the DARPA Agent Markup Language (DAML). In addition to keyword-matching, the WSPDS query engine also supports semantic-matching of the operational service interfaces using the matchmaking algorithm proposed in [38]. Each QDN virtual node represents a service operation; one physical node can carry several virtual QDN nodes. The identity of QDN virtual nodes is defined as the ontologies associated with the input/output service parameters. When a node joins the network it is linked to the nodes that have semantically the most similar input/output. In query propagation, each servent that receives a query forwards it to the neighbor with the most similar identity to the query. A related approach is presented in [11] where overlay network topology is inspired by small-world graphs with node proximity defined according to their semantic similarity. Similarly in [?] authors build a CAN-based document repository network where CAN keys are generated according to the so-called *Latent Semantic Indexing* (LSI).

The scalable semantic routing architecture for Grid service discovery, presented in [30], goes beyond [26] in the sense that it utilizes a hierarchical structure to improve the scalability and robustness. The Resource Description Framework (RDF) is used to represent both resources and queries. A hierarchical routing algorithm supports complex queries without resorting to network flooding. The routing algorithms use Bloom filters for aggregation resource information and to help route the queries. Experimental results prove the efficiency and scalability of the scheme. In [35] a related approach called *schema-based networks* is presented, which is based on super-peer topologies.

Distributed hash tables can be used for storing semantic information in Grids. For example, in [22], the authors present an approach to semantic resource discovery in the Grid. A P2P network maintains a resource catalogue using DHT algorithms. Peers provide resource descriptions and background knowledge in ontologies based on description logic, and each peer can query the network for available resources. Each peer may have its own ontology represented as a *classification DAG* (Directed Acyclic Graph), which captures subsumption relations between concepts in the ontology. The peer's ontology is possibly incomplete, but it can be completed by ontologies of other peers. The authors propose a DHT algorithm that distributes local classification DAGs among nodes of the P2P network to form a distributed virtual classification DAG used by all peers for resource discovery. The DHT algorithm uses a concept name as a key to determine the node which will store the information for this concept including a list of super-concepts according to the DAG and a list of resources – instances of the concept. When querying for a simple concept, the DHT is looked up using the concept name to determine the node which is responsible for the concept being queried. A complex query is formulated in terms of simple concepts, and the discovery result is built up by querying the network for instances of every simple concept which occurs in the complex concept. When a new node joins the network, it iterates through its classification DAG to identify the nodes which need to be informed about new super-concepts or new instances. The nodes collaborate to disseminate the new information in the network. This way, the distributed ontology grows as more and more peers joining the network publish their local knowledge.

When a node leaves the network, a dedicated distributed algorithm removes semantic information on concept instances for which the node was responsible. Simulation results demonstrate that this approach scales well for large number of concepts and nodes.

Another ontology-based search scheme using DHTs is proposed in [44]. *Semantic-aware network* (*SA Net*) is a structured P2P overlay architecture that supports basic functionalities of personalized resource discovery. Semantic resource discovery reflecting user interests is enabled by ontology-based resource representations. In the paper, the SA Net search scheme called *Semantic-driven Hashing* (*SDH*) is presented. SDH uses lexical-based ontology that allows indexing and searching in structured P2P overlay infrastructure.

A two-level service discovery architecture is proposed in [57]. The lower level organizes community overlays of different service categories defined in the service ontology. Message propagation is limited into related communities only. The upper layer is based on a DHT which provides efficient navigation between communities. The intra-community search for service providers is based on a simple and lightweight greedy search based service location (*GSBSL*) method. Simulation results show that while the search efficiency is improved compared to flat overlays, the management overhead is still acceptable and controllable. A related approach [29] builds a SkipNet-based [20] category overlay organizing unstructured communities of semantically-similar nodes.

5 Final remarks and future research directions

P2P has emerged as a successful paradigm in distributed computing thanks to its inherent scalability and robustness, which promises to enable the development of global-scale, cooperative, distributed applications. This is also witnessed by the fact that several P2P systems for resource discovery in Grid environments have been recently proposed. Such systems adopt different models and solutions, including structured or unstructured overlay networks, fully decentralized or super-peer architectures, and diverse strategies for improving routing performance and search precision. Moreover, they provide very different search capabilities, ranging from single-attribute search to multi-attribute and range queries.

An important aspect that distinguishes Grid from P2P systems is the organization of resources. As opposed to P2P systems, large-scale Grids are generally built as federations of smaller Grids managed by diverse organizations. This organization-based architecture applies to most of the systems discussed earlier in this paper, in which typically one node per organization participates in the P2P network [23,51,33,39]. Supporting very dynamic environments is fundamental, since the availability and status of resources within each node change dynamically over time. Another fundamental requirement in Grid systems is the ability to perform multi-attribute and range queries. In what follows we discuss the ability of each of the main types of P2P systems, unstructured, structured, and hybrid, in relation to the basic requirements which have been introduced in Section 2: scalability, reliability, and support for dynamicity.

With respect to scalability, structured systems perform better than unstructured systems, since Distributed Hash Tables (DHTs) are more scalable, self-organizing and load balanced than pure-P2P overlay networks. Another important advantage of DHTs is their ability to efficiently support range queries inherited from their data locality property. All structured systems described in this paper provide support for range queries, and most of them also provide multi-attribute search capabilities [8,36,47]. On the other hand, structured systems can be more difficult to maintain in very dynamic Grid environments, where the availability and status of resources vary significantly over time. As discussed in Section 2, for each resource in the system, the peer with the appropriate ID must be notified periodically, resulting to either increased traffic (if the period is too small), or stale information (if the period is too large). Unstructured systems, on the other hand, adopt diverse strategies to provide up-to-date results with limited network traffic, including experience-based query forwarding [23,33], message buffering and merging [51], routing indexes [39,32], and super-peer architectures [33,39]. In particular, some experiments [33] show that the super-peer model is naturally appropriate to the organization-based nature of current Grids, ensuring limited network load and reduced response time with respect to pure-decentralized P2P systems. As stated in Section 2, both unstructured and structured systems show advantages and disadvantages. Hybrid P2P approaches can be adopted to combine the efficiency of structured systems and the dynamicity of unstructured system, while overcoming their inherent drawbacks. For instance, structured protocols could be adopted for relatively static information, whereas unstructured approaches could be employed for more dynamic information. Moreover, the organization-based nature of Grids suggests the use of a super-peer architecture, in which different strategies (e.g. structured or unstructured protocols) may be adopted for intra-organization and inter-organization resource discovery. Finally, the OGSA model can be of great importance to federate different information services into a Grid resource discovery system, using Web services conventions as a means to ensure interoperability among the various peer subsystems [51,33,39].

Another aspect of the problem is to extend resource descriptions with semantic annotations that give well-defined meaning to resource information to better enable VOs to cooperate in resource discovery. Adding semantic information to resource descriptions also allows improving precision of a discovery service that should be able to find best approximate matches usable for the requester as it is unrealistic to expect requested and offered services to be exactly identical. Even though the use of semantic information in resource discovery is very important for interoperability, it raises a problem of *semantic interoperability*, i.e. it requires using common ontologies in service descriptions in order to reach semantic agreement. Using semantic information for precise resource discovery in large-scale, dynamic and heterogeneous environments is a relatively new and fragmented research topic. We believe that more studies should be devoted to comparing relative merits of proposed approaches and architectures. We also believe that approaches to scalable resource discovery in P2P systems can be useful for building scalable semantic-based resource discovery in Grids, in particular, for building a distributed knowledge-base for resources descriptions as well as a distributed storage for ontologies. We envision that resource discovery services that use P2P-based networks for scalable and reliable storage of semantic information can appear soon in standard Grid middleware such as Globus and gLite. The distributed matchmaking and semantic-based routing promise improvements in discovery precision and cost. Also, P2P-based service composition/workflow construction using semantic service descriptions should be studied from the resource discovery respective, i.e. as a form of resource discovery.

References

[1] eDonkey2000. http://edonkey2000.com

[2] Dynamic Query Protocol. http://www.the-gdf.org/wiki/index.php?title=Dynamic_Query_Protocol

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

[4] GUESS Protocol Specification.

[5] Napster. http://www.napster.com

[6] A. Andrzejak and Z. Xu, "Scalable, Efficient Range Queries for Grid Information Services", *Proc. 2nd Int. Conf. on Peer-to-Peer Computing (P2P 2002)*, pp. 33-40, 2002.

[7] A.R. Bharambe, M. Agrawal and S. Seshan, "Mercury: Supporting Scalable Multi-Attribute Range Queries", *Proc. ACM SIGCOMM 2004 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 353-366, 2004.

[8] M. Cai, M. Frank, J. Chen and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services". *Proc. 4th Int. Workshop on Grid Computing (GRID 2003)*, pp. 184-191, 2003.

[9] M. Cannataro and D. Talia, "Semantics and Knowledge Grids: Building the Next-Generation Grid", *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 56-63, 2004.

[10] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker, "Making Gnutella-like P2P Systems Scalable". *Proc. ACM SIGCOMM 2003 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 407-418, 2003.

[11] Hanhua Chen, Hai Jin, Xiaoming Ning, "Semantic Peer-to-Peer Overlay for Efficient Content Locating". *Proc. Int. Work. on Advanced Web and Network Technologies (APWeb 2006)*, pp. 545–554, 2006.

[12] A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems". *Proc. 22nd Int. Conf. on Distributed Computing Systems (ICDCS'02)*, pp. 23-30, 2002.

[13] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems". Technical Report, Stanford University, 2003.

[14] F. Dabek, E. Brunskill, M. Frans Kaashoek, D.R. Karger, R. Morris, I. Stoica and H. Balakrishnan, "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service". *Proc. 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pp. 81-86, 2001.

[15] D. De Roure, N. R. Jennings and N. Shadbolt, "The Semantic Grid: Past, Present and Future". *Proc. IEEE*, 93, 2005.

[16] M. Frans Kaashoek and D.R. Karger, "Koorde: A Simple Degree-optimal Distributed Hash Table". *Proc. Second Int. Workshop on Peer-to-Peer Systems (IPTPS 2003)*, pp. 98-107, 2003.

[17] I.T. Foster, N.R. Jennings and C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other". *Proc. Third Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, 2004.

[18] C. Gkantsidis, M. Mihail and A. Saberi, "Hybrid Search Schemes for Unstructured Peer-to-Peer Networks". *Proc. IEEE INFOCOM*, 2005.

[19] C.A. Goble and D. De Roure, "The Semantic Grid: Myth Busting and Bridge Building". *Proc. 16th European Conf. on Artificial Intelligence (ECAI-2004)*, pp. 1129-1135, 2004.

[20] N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties", *Proc. USENIX Symp. on Internet Technologies and Systems*, 2003.

[21] D. Heimbigner, "Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality". *Proc. 2001 ACM Symposium on Applied Computing (SAC)*, pp. 176-181, 2001.

[22] F. Heine, M. Hovestadt, and O. Kao. Towards ontology-driven P2P Grid resource discovery. In *5th IEEE/ACM International Workshop on Grid Computing*, November 2004.

[23] A. Iamnitchi and I.T. Foster, "A Peer-to-Peer Approach to Resource Location in Grid Environments", In: J. Weglarz, J. Nabrzyski, J. Schopf and M. Stroinski (Eds.), *Grid Resource Management*, Kluwer, 2003.

[24] A. Iamnitchi and D. Talia, "P2P computing and interaction with grids", *Future Generation Computer Systems*, vol 21, no. 3, pp. 331-332, 2005.

[25] B. Iannucci, "Connected Lifestyles: The Next Big Wave", Infotech day, November 2005, University of Oulu.
http://www.infotech.oulu.fi/paiva/2005/bob_iannucci.pdf

[26] F.B. Kashani, C.C. Chen and C. Shahabi, "WSPDS: Web Services Peer-to-Peer Discovery Service". *Proc. Int. Conf. on Internet Computing (IC'04)*, 2004.

[27] F.B. Kashani and C. Shahabi, "Searchable Querical Data Networks". *Proc. First Int. Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P), LNCS*, vol. 2944, pp. 17-32, Springer 2003.

[28] M. Li, P. van Santen, D.W. Walker, O.F. Rana and M.A. Baker, "SGrid: A Service-Oriented Model for the Semantic Grid". *Future Generation Computer Systems (FGCS)*, vol. 20, no. 1, pp. 7-18, 2004.

[29] Juan Li and Son Vuong. "Grid Resource Discovery Based on Semantic P2P Communities". *Proc. ACM Symp. Applied Computing (SAC2006)*, pp. 754-758, 2006.

[30] J. Li and S. Vuong, "A Scalable Semantic Routing Architecture for Grid Resource Discovery". *Proc. 11th Int. Conf. on Parallel and Distributed Systems (ICPADS'05)*, vol. 1, pp. 29-35, 2005.

[31] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks". *Proc. 16th Annual ACM Int. Conf. on Supercomputing (ICS 2002)*, pp. 84-95, 2002.

[32] M. Marzolla, M. Mordacchini and S. Orlando, "Resource Discovery in a Dynamic Grid Environment", *Proc. DEXA Workshop 2005*, pp. 356-360, 2005.

[33] C. Mastroianni, D. Talia and O. Verta, "A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis". *Proc. European Grid Conference (EGC 2005)*, LNCS, vol. 3470, pp. 132-143, Springer 2005.

[34] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". *Proc. First Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 53-65, 2002.

[35] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser, "Super-peer-based Routing and Clustering Strategies for RDF-based Peer-to-peer Networks. *Proc. 12th Int. Conf. World Wide Web (WWW '03)*, pp. 536–543, 2003.

[36] D. Oppenheimer, J. Albrecht, D. Patterson and A. Vahdat, "Scalable Wide-Area Resource Discovery". TR CSD04-1334, Univ. of California, 2004.

[37] C. Papadakis, P. Fragopoulou, E. Athanasopoulos, M. Dikaiakos, A. Labrinidis and E. Markatos, "A Feedback-based Approach to Reduce Duplicate Messages in Unstructured Peer-to-Peer Networks", *Integrated Workshop on Grid Research*, 2005.

[38] M. Paolucci, T. Kawamura, T.R. Payne and K.P. Sycara, "Semantic Matching of Web Services Capabilities". *Proc. First Int. Semantic Web Conf. on The Semantic Web*, pp. 333-347, 2002. Springer-Verlag.

[39] D. Puppin, S. Moncelli, R. Baraglia, N. Tonelotto and F. Silvestri, "A Grid Information Service Based on Peer-to-Peer". *Proc. 11th Euro-Par Conf. (Euro-Per 2005)*, LNCS, vol. 3648, pp. 454-464, Springer 2005.

[40] S. Ratnasamy, P. Francis, M. Handley, R.M. Karp and S. Shenker, "A Scalable Content-Addressable Network". *Proc. ACM SIGCOMM 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 161-172, 2001.

[41] S. Ratnasamy, J.M. Hellerstein and S. Shenker, "Range Queries over DHTs", IRB-TR-03-009, Intel Corporation 2003.

[42] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems". *Proc. IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware 2001)*, pp. 329-350, 2001, LNCS, vol 2218, Springer 2001.

[43] M. Sanchez, P. Garcia, J. Pujol, A. Skarkemta, "Cyclone: a Novel Design Schema for Hierarchical DHTs". *Proc. Fifth IEEE Int. Conf. Peer-to-Peer Computing (P2P 2005)*, pp. 49-56, 2005

[44] C. Sangpachatanaruk and T. Znati, "Semantic Driven Hashing (SDH): An Ontology-Based Search Scheme for the Semantic Aware Network (SA Net)". *Proc. Fourth Int. Conf. on Peer-to-Peer Computing (P2P'04)*, pp. 270-271, 2004.

[45] C. Schmidt and M. Parashar, "Flexible Information Discovery in Decentralized Distributed Systems", *Proc. 12th Int. Symp. on High-Performance Distributed Computing (HPDC-12 2003)*, pp. 226-235, 2003.

[46] A. Singla and C. Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability".
http://rfc-gnutella.sourceforge.net/src/Ultrapeers_1.0.html

[47] D. Spence and T. Harris, "XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform", *Proc. Twelfth IEEE Int. Symposium on High Performance Distributed Computing (HPDC-12)*, pp. 216-225, 2003.

[48] K. Sripanidkulchai, B. Maggs and H. Zhang, "Efficient Content Location using Interest-Based Locality in Peer-to-Peer Systems". *Proc. IEEE INFOCOM*, 2003.

[49] I. Stoica, R. Morris, D.R. Karger, M.Frans Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications". *Proc. ACM SIGCOMM 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 149-160, 2001.

[50] D. Talia and P. Trunfio, "Toward a Synergy between P2P and Grids". *IEEE Internet Computing*, vol. 7, no. 4, pp. 94-96, 2003.

[51] D. Talia and P. Trunfio, "Peer-to-Peer Protocols and Grid Services for Resource Discovery on Grids". In: L. Grandinetti (Ed.), *Grid Computing: The New Frontier of High Performance Computing, Advances in Parallel Computing*, vol. 14, Elsevier Science, 2005.

[52] D. Tsoumakos and N. Roussopoulos, "A Comparison of Peer-to-Peer Search Methods". *Proc. Int. Workshop on Web and Databases (WebDB 2003)*, pp.61-66, 2003.

[53] G. Wepiwe and P.L. Simeonov, "HiPeer: A High Reliable P2P System". *IEICE Tans. Inf. & Syst.*, vol. E98-D, no. 2, pp. 570-58-, 2006.

[54] B. Yang and H. Garcia-Molina, "Designing a Super-Peer Network". *Proc. Int. Conference on Data Engineering (ICDE 2003)*, pp. 49-60, 2003.

[55] D. Zeinalipour-Yazti, V. Kalogeraki and D. Gunopulos, "Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Systems". *Information Systems J.*, vol. 30, no. 4, pp. 277-298, 2005.

[56] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment". *IEEE J. on Selected Areas in Communications*, vol. 22, no. 1, pp. 41-53, 2004.

[57] C. Zhu, Z. Liu, W. Zhang, W. Xiao and J. Huang, "An Efficient Decentralized Grid Service Discovery Approach Based on Service Ontology". *Proc. Web Intelligence, IEEE/WIC/ACM Int. Conf. (WI'04)*, pp. 570-573, 2004.