Resource discovery mechanisms in grid systems: A survey

Article *in* Journal of Network and Computer Applications · May 2014

Impact Factor: 2.23 · DOI: 10.1016/j.jnca.2013.09.013

4 authors:

Nima Jafari Navimipour
Islamic Azad University Tabriz Branch
22 PUBLICATIONS **111** CITATIONS
SEE PROFILE

Amir Masoud Rahmani
Islamic Azad University Tehran Science and…
199 PUBLICATIONS **648** CITATIONS
SEE PROFILE

Ahmad Habibizad Navin
Islamic Azad University
48 PUBLICATIONS **197** CITATIONS
SEE PROFILE

Mehdi Hosseinzadeh
Islamic Azad University
31 PUBLICATIONS **102** CITATIONS
SEE PROFILE

Journal of Network and Computer Applications 41 (2014) 389–410

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Resource discovery mechanisms in grid systems: A survey

Nima Jafari Navimipour [b], Amir Masoud Rahmani [a], Ahmad Habibizad Navin [b,*], Mehdi Hosseinzadeh [a]

[a] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Department of Computer Engineering, East Azarbaijan Science and Research Branch, Islamic Azad University, Tabriz, Iran

ARTICLE INFO

Article history:
Received 12 April 2013
Received in revised form
21 July 2013
Accepted 30 September 2013
Available online 10 October 2013

Keywords:
Grid systems
Peer to peer
Resource discovery
Network applications
Scalability
Centralized
Decentralized
Hierarchical
Agent

ABSTRACT

Grid computing systems provide a vast amount of computing resources from multiple administrative domains to reach a main objective. One of the most important challenges in these systems is to discover appropriate resource in networks. In this paper we survey the resource discovery mechanisms which have been used in Grid computing systems so far. We classify the resource discovery mechanisms into five main categories: Centralized, Decentralized, Peer to Peer, Hierarchical, and Agent based. We reviewed the major development in these five categories and outlined new challenges. This survey paper also provides a discussion of differences between considered mechanisms in terms of scalability, dynamicity, reliability and queries' attributes as well as directions for future research.

Contents

* Corresponding author. Tel.: +98 9144125973; fax: +98 4113817221.
E-mail addresses: n.jafari@srbiau.ac.ir (N. Jafari Navimipour), rahmani@srbiau.ac.ir (A. Masoud Rahmani), a.habibizad@srbiau.ac.ir (A. Habibizad Navin), Hosseinzadeh@srbiau.ac.ir (M. Hosseinzadeh).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.09.013

1. Introduction

The term "Grid" was coined in the mid 1990s offering access to a vast collection of heterogeneous resources as a single, unified resource to solve large-scale computing and data intensive problems for advanced science and engineering (Balasangameshwara and Raju, 2012; Erdil, 2012; Ian Foster and Kesselman, 1999; Siva Sathya and Syam Babu, 2010). Some examples of Grid computing systems are NASA IPG (Johnston et al., 1999), the World Wide Grid,[1] HealthGrid (Naseer and Stergioulas, 2010), and Selenium-Grid.[2] After almost 20 years of development, Grid computing has made many varieties, such as computational Grid which denotes systems that have higher aggregated computational capacity available for single applications; data Grid which provides an infrastructure for synthesizing new information from data repositories such as digital libraries or data warehouses that are distributed in a wide area network; information Grid which is the integration of information across heterogeneous data sources; service Grid which provides services that cannot be provided by any single machine; wireless Grid which enables wireless and mobile users to share computing resources, services, and information (Moreno-Vozmediano, 2009); A multimedia Grid which provides an infrastructure for real-time multimedia applications and cloud computing (Krauter et al., 2002).

The Grid architecture adopts a layered structure that corresponds to the Internet protocol architecture (Fig. 1). The Grid architecture consists of four layers: fabric, connectivity, collective and application layer. The fabric layer in this structure refers to a set of resources or devices, including computers, storage systems, networks, and various types of computer-controlled instruments and devices. The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions (Foster et al., 2001). The Resource layer is built on Connectivity layer communication and authentication protocols to define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources (Foster et al., 2001). Above the resource and connectivity layers, the collective layer contains protocols, services, and APIs that implement interactions across collections of resources, therefore collective layer coordinates multiple resources (Wang et al., 2009) that provide facilities to access some useful services such as resource discovery and management. The final layer in Grid architecture is the application layer which comprises the user applications (Foster et al., 2001).

In Grid, there are many types of resources such as computers, cluster of computers, online tools, storage space, data, and applications (Iamnitchi and Foster, 2001) which are widely distributed and heterogeneous in comparison to traditional and cluster systems. Resource discovery is one of the essential challenges in Grid, which discovers appropriate resources based on the requested task. There are certain factors that make the resource discovery problem difficult to solve. These factors are the huge number of resources, distributed ownership, heterogeneity of resources, resource failure, reliability, dynamicity and resource evolution (Hameurlain et al., 2010). These factors are essential criteria for designing a good resource discovery mechanism.

In this paper, we divided most of the introduced resource discovery algorithms into five main categories, centralized, decentralized, peer to peer, hierarchical, and agent-based. This paper provides a survey on resource and service discovery mechanisms in Grid systems and compares the differences between mentioned mechanisms and describes several popular resource discovery mechanisms. Also a taxonomy to differentiate between considered mechanisms is provided.

The rest of this paper is structured as follows. The basic concepts and terminologies are provided in the next section. Section 3 discusses resource discovery mechanisms in Grid system and categorizes them. Section 4 presents the taxonomy and comparison of proposed mechanisms. Section 5 maps out some open issues. Finally Section 5 concludes this paper.

2. Basic concepts and related terminologies

This section introduces the basic concepts and related terminologies which are used in this paper. We explain the following concepts and terminologies:

Scalability: scalability is one of the important issues for designing resource discovery mechanisms. It defines the ability of a resource discovery mechanism to handle a growing amount of Grid systems with predefined level of efficiency.

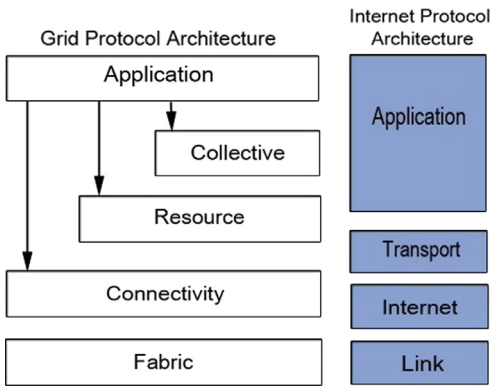

Fig. 1. Grid architecture compared with internet architecture (Ian Foster et al., 2001).

[1] http://www.buyya.com/ecogrid/wwg/.
[2] www.selenium-grid.seleniumhq.org.

Multi-Attribute Query: a multi-attribute query is a set of sub-queries, in which each subquery involves an attribute. The result of the query is an intersection of all results from all subqueries (Jagadish et al., 2006).

Static/Dynamic Attribute Query: Each query in Grid systems is either static or dynamic attribute. Intuitively, a static attribute is an attribute that changes only when an explicit update of the database occurs; in contrast, a dynamic attribute changes over time according to some given function, even if it is not explicitly updated (Sistla et al., 1997).

Periodic Update of Resource information: Periodic update of resource information is the update that occurs periodically. Because of periodical update and the changes in attribute of the query, it limits the utility of dynamic attribute queries (Cokuslu et al., 2010).

Range Queries: Range queries involve numeric (or numerical) attributes. These are attributes whose domain is totally ordered and thus a query interval can be formed. Range queries correspond to selections and are thus amenable to indexing (Moro, 2009).

Bottleneck: A bottleneck is a situation where the performance of a resource discovery mechanism is decreased by a single or limited number of discovery components.

Single Point of Failure: The single point of failure is any part within a resource discovery process whose failure leads to the failure of the discovery mechanism.

Load Balancing: Load balancing in a Grid is a hot research issue which affects every aspect of the Grid, including service selection and task execution. The load balancing algorithms attempt to improve the response time of a user's submitted applications by ensuring maximal utilization of available resources (Hao et al., 2012).

Fault Tolerance: Fault tolerance is the ability of a resource discovery mechanism to continue valid operation or find other resources after some resources, application, or part of them, fail in some way.

Security: Security is a vital part of the integrated Grid systems in which heterogeneous services and resources belonging to multiple domains are distributed dynamically (Ruckmani and Sadasivam, 2010). Also, security issues (authentication, encryption, and authorization) are obviously a major concern when users share their resources and information in a Grid environment (Moreno-Vozmediano, 2009). Therefore resource discovery mechanism should try to provide high level of security. One of the main problems of resource discovery is to face the different security profiles of nodes belonging to different administrative domains. Indeed, each node has its own security manager that impedes access to any other that is not certified (Di Stefano et al., 2009).

Node Dynamicity: Node dynamicity means that any node (resource or service) in Grid systems can joins, leaves or fails at any time (Hameurlain, 2009) and also it can changes its characteristics.

Reliability: The reliability of the Grid systems is estimated by focusing on the reliabilities of services provided by service providers (Doguc and Emmanuel Ramirez-Marquez, 2012). Also it is defined as the probability that a set of programs contained by a grid service can be successfully completed (Horng, 2011).

Flexibility: The flexibility is the ability of a resource discovery mechanism to respond to potential internal or external modifications in a timely and cost-effective manner. The method of a resource discovery mechanism should be flexible for the ease of modification, and the ease of enhancement.

Distributed Hash Table (*DHT*): A DHT is a distributed system that efficiently maps “keys” to “values”, and efficiently routes queries about information to the unique owner of the key related to that information. The mapping of information to numeric keys is done using a hash function (Belqasmi et al., 2011).

False-positive error: It is a result that indicates a resource discovery process has been fulfilled successfully, when there is a chance that a resource is not found even if present. The usage of TTL causes false-positive errors in most of unstructured systems.

3. Resource discovery mechanisms

A fundamental service in the Grid computing systems is resource discovery which finds the appropriate resources for requested task matching the user's application requirements (Sarhadi et al., 2012). Also this problem can be defined as searching and locating resource candidates which are suitable for executing jobs in a reasonable time in spite of dynamicity and large scale of the environment (Cokuslu et al., 2010). Allocating resources on Grid computing systems is a complex procedure involving dynamic and multi-attribute queries, sharing and meeting the requirements of users and resource owners (Vanderster et al., 2009). In this section, some important resource discovery mechanisms are reviewed and classified into five categories which are illustrated in Fig. 2 and discussed as follows.

3.1. Centralized mechanisms

In this section, we first describe the centralized mechanisms of resource discovery and their basic properties. Second, we discuss five most popular centralized mechanisms of resource discovery. Finally, the discussed centralized mechanisms are compared and summarized in Section 3.1.3.

3.1.1. Overview of centralized mechanisms

In the centralized mechanisms a single or designated set of controllers discovers the resources which follow client/server architecture (Krauter et al., 2002). In these mechanisms, the servers store information about the services which can be provided. When an entity requests a certain service, it sends a request to the server, and then finds appropriate resources and allocates them to the requester's entity. Since all query processes are done by a single or designated set of controllers, once a system exceeds several hundred nodes, the resource discovery mechanism becomes a bottleneck. However, these mechanisms typically have the fastest search time. The next section provides a review and survey on several important centralized mechanisms of resource discovery.

3.1.2. Popular centralized mechanisms

In this section, some popular and applicable centralized mechanisms of discovering appropriate resources in Grid environments are discussed.

A Meta-computing Directory Service (MDS) for resource management in Grid systems was proposed by Fitzgerald et al. (1997). MDS uses the data representation and application programming interface (API) and is maintained by central servers based on Lightweight Directory Access Protocol (LDAP). In MDS, resources are represented by MDS entries in the LDAP server which are specialized data structures. Resource information maintenance and responses to the

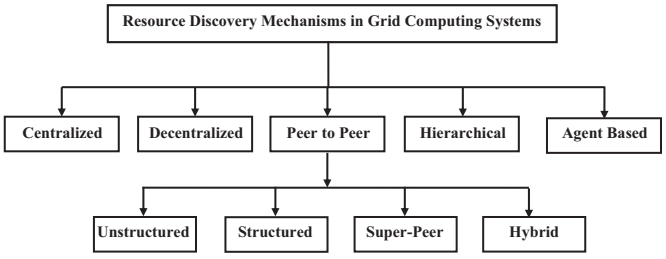

Fig. 2. Resource discovery mechanisms classification in Grid computing systems.

queries are handled by LDAP servers. The proposed method provides efficient access to diverse, dynamic, and distributed information about resource structure and states, but it has single point of failure because of its centralized nature of MDS; therefore, the initial MDS design was developed and enhanced (MDS-2) to work in a hierarchical manner in Czajkowski et al. (2001) which will be discussed in Section 3.4.

Grid Market Directory (GMD) which is a web service based on Grid service publication directory which provides service for resources and clients via web by using XML formatted messages which were presented by Yu et al. (2003). The GMD was more completed than MDS as it provides higher-level services and designed to enable the idea of Grid economy. The GMD consists of two main components; GMD Portal Manager and GMD Query Web Service. GMD Portal Manager provides a web-based platform to manage registration and publication of resources. A user uses the portal to advertise its resources to the central repository in order to share its resources. GMD Query Web Service is a query processing web service which is used to discover required resources that meet the query requirements (Yu et al., 2003). GMD uses a central resource repository database and central query processing engine. Therefore, the system may suffer from bottleneck and single point of failure in a large scale Grid system. Moreover, it is not suitable for dynamic Grid environments since the dynamicity requires manual updates. On the other hand, the system supports multi-attribute and range queries since the repository is a database system in which those types of queries can easily be supported. However, it does not support dynamic-attribute queries. Also it has high response time in high workload.

Kaur and Sengupta (2007) presented a centralized resource discovery mechanism in web-services based Grids. The proposed system consists of four main components; Universal Description Discovery and Integration (UDDI) (Bellwood, 2002), Grid Web Services Description Language (GWSDL) (Christensen et al., 2001), Simple Object Access Protocol (SOAP) (SOAP Version 1.2 Part 0: Primer, 2007) and Hypertext Transfer Protocol (HTTP). UDDI is an open standard for publishing and discovering the software components of service-oriented architecture. Proposed mechanism uses UDDI standard (Benson et al., 2006) to discover Grid services and hold Grid resource information as key-value pairs in the UDDI database. Furthermore, Grid Web Service Description Language (GWSDL) is used to describe Grid services in an extended version of Web Services Description Language. SOAP establishes communication between web services in the Grid environment. HTTP provides the interface to send requests and get responses (Kaur and Sengupta, 2007). The system uses central servers and databases to run the web service. Therefore, it contains bottlenecks and a single point of failures. Moreover, since the resource information is held in a repository database, dynamic attribute queries are not supported. But, the proposed mechanism supports range and multi-attribute queries. In addition, using UDDI for Grid service has some problems because it is designed to be used for business services. For example, missing of explicit data type in UDDI directory; difficulties in handling regularly updated dynamic information such as continuous numeric type of CPU load which changes at instances; and limited query capability. Also the scalability of the mechanisms is very low.

To overcome the problems of UDDI Benson et al. (2006) have evaluated internal structure of it to discover OGSA[3]-based Grid services as a provider of resource discovery services in 2006. The proposed mechanism is a new UDDI centralized model of Grid resource discovery with the following modifications: the issue of explicit data type, which did not exist in UDDI registry, is resolved by proposing the continuous variables of numeric type in UDDI registry; the issue of dynamic information is resolved by introducing a new variable called Last-Update-Time in the UDDI registry to store periodic update from resource providers; and the issue of limited query model is resolved by associating performance data like CPU load or machine attributes with a reference key (Benson et al., 2006). This mechanism provides its functionality through four principle entities: the businessEntity which contains information about any organizational unit; businessService which provides information about the service name, categorization, and any other useful details added in a variety of attribute lists; tModel which is the standard way to describe specific details about a particular businessService or bindingTemplate in UDDI; and the bindingTemplate which represents the link between abstract businessService descriptions and actual endpoints at which these services may be accessed (Benson et al., 2006). Consequently, UDDI can be used in Grid service discovery but with the above mentioned modifications. Also, experiments have been done to find the performance of a mechanism under the system load measured by the update frequency of Grid resources. Since the mechanism is administered from a central system the security of the system can be high. However, the mechanism suffers from bottlenecks and dynamic attribute queries are not supported. Also the scalability of the mechanisms is very low.

Kovvur et al. (2010) presented the adaptive resource discovery and resource selection models in Grid. They discussed three models: Adaptive pull model, Adaptive push model, and Adaptive push-pull model. In adaptive pull model, Grid environment consists of several nodes, out of which one of the node is made as coordinator where a single daemon is running. This daemon can pull and collect dynamic state of information such as CPU speed, CPU loads, and memory size from various remote nodes. The traffic to daemon is very high and it is a single point of failure. Also because of high traffic to daemon, as the Grid environment becomes larger and larger, the pull resource discovery query process would begin taking significant amounts of time. In adaptive push model, Grid environment consists of several nodes, out of which one of the node is used as coordinator, where a master daemon is running and other nodes in the environment has its own daemon for gathering local state information, which will be pushed to a coordinator. The centralized coordinator maintains a database to record each resource activity. Obviously, frequent and periodic updates to the database are intrusive and consume network bandwidth. Also master daemon and coordinator still are single point of failures and endure high traffic. The push–pull model lies somewhere between the pull model and the push model .The Grid environment in this model consists of three layers: Main coordinator (layer-1), Aggregators (layer-2) and Grid nodes (layer-3). Each Grid node runs a daemon that collects state information such as CPU speed, CPU loads, and memory size. This collected state information will be pushed to the Aggregators, instead of directly being sent to a main coordinator. Each Aggregator maintains a database of aggregate state information of these sub resources. The daemon at main coordinator pulls the aggregate state information from the Aggregator and respond to queries. This model also has single point of failure in the coordinator, but high traffic to the coordinator is decreased (Kovvur et al., 2010). These models support multi-attribute queries but in large scale Grid is very inefficient.

3.1.3. Summary of centralized mechanisms

The discussed centralized mechanisms provide facility to access Grid services; in which resource information is stored and updated in central servers. Most of these mechanisms can support multi-attribute queries but because of network traffic, dynamic attribute queries are not supported. The authorization and security issues

[3] Open Grid Services Architecture.

Table 1
Popular centralized mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
MDS (Fitzgerald et al.)	Defining an extensible data model to represent required information	Efficient access to diverse, dynamic, and distributed information	Contains bottlenecks, dynamic and multi-attribute queries are not supported
GMD (Yu et al.)	Providing service for resources and clients via web by using XML formatted messages	Completes the MDS by enabling the idea of Grid economy and supports multi-attribute queries	Contains bottlenecks; dynamic attribute queries are not supported; has high response time in high workload
Kaur and Sengupta	Performing the resource discovery and presenting the key components of web-service based Grids	Supports multi-attribute queries	Contains bottlenecks; dynamic attribute queries are not supported; scalability is low
Benson et al.	Modifies UDDI and presents a new centralized model based on it, and provides its functionality through four principle entities	Supports multi-attribute queries; provides high security	Contains bottlenecks; dynamic attribute queries are not supported; scalability is low
Kovvur et al	The authors discussed three model: Adaptive pull, Adaptive push, and Adaptive push-pull model	Supports multi-attribute queries	Contains bottlenecks, a single point of failure and high traffic to some nodes

are handled quite easily as they are administrated from a central point. The main disadvantage of these mechanisms is that they do not scale well when frequent updates and large numbers of requests exist. When a system has several hundred nodes, a bottleneck is created in the server which runs the RD service. This is especially true when computationally complex queries are executed. Furthermore, parallel accesses to the central database are very limited and the response time could be very high for a high workload. In order to overcome these drawbacks, the global database must be replicated to prevent the case of failure (Ludwig, 2003), this idea is the basis of decentralized mechanisms which are discussed in the next section.

Table 1 summarizes the discussed centralized mechanisms and introduces their advantages and disadvantages.

3.2. Decentralized mechanism

In this section, the decentralized mechanisms of resource discovery and their basic properties are described. Then, seven most popular decentralized mechanisms of resource discovery are discussed. Finally, these mechanisms are compared and summarized in Section 3.2.3.

3.2.1. Overview of decentralized mechanisms

As discussed in the previous section the centralized resource discovery mechanism is not suitable for large-scale networks. There are alternative resource discovery mechanisms studied by many researchers. In these mechanisms, the central database or server has been removed and all nodes work together to do the resource discovery operation in the large-scale system. These mechanisms act better than centralized mechanism in a large scale Grid but additional overhead is created by managing the network architecture.

3.2.2. Popular decentralized mechanisms

Fully decentralized resource discovery in Grid environments was proposed by Iamnitchi and Foster (2001). They proposed a flat, decentralized, self-configuring architecture, where resources are located on network nodes. A user connects to a local node and the resource discovery is performed by forwarding the request node by node. The node either responds with the matching resource or forwards the request to another node. The request is being forwarded until a resource is found or the initial time-to-live (TTL) value in the request message is decreased to zero (Iamnitchi and Foster, 2001). A node can forward a request using one of the four request forwarding algorithms: "random", "experience-based + random", "best-neighbor", and "experience-based + best-neighbor". The authors showed that "experience-based + random-algorithm" gives the best performance among four algorithms. Also all mechanisms are based on First-Found-First-Served (FFFS) algorithm. But FFFS is insufficient for the resource discovery mechanism in Grid environments as the first resource with desired attributes is allocated to requester without more searching (Tangpongprasit et al., 2005). Also, due to the large amount of query messages generated by flooding, this mechanism does not have high scalability. However, due to random-walk based methods are used for query forwarding, this mechanism are inefficient in response time for a very large system. In addition, the proposed mechanism cannot guarantee that requested resources can always be found even if they exist in the system (false-positive error). But this mechanism provides good load balancing.

Zhu et al. (2004) presented a decentralized Grid resource discovery based on resource information communities in which resources are discovered according to their attributes rather than their identifiers. In this mechanism information nodes with the same type of resources are grouped to resource information communities, and efficient navigation is supported via a Distributed Hash Table (DHT) based bootstrap network. By limiting searching and resource information propagation within related communities, the author shows that the performance of Grid resource discovery is significantly improved while the topology maintenance overhead was increased. Moreover, constraints on relations between different resources in discovery process were not considered.

A flat decentralized resource discovery algorithm with TTL-based reservation and unicast request forwarding algorithms on a flat and fully decentralized architecture in Grid computing was proposed by Tangpongprasit et al. (2005). The authors presented a new algorithm that differs from FFFS which is presented by Iamnitchi and Foster (2001). The proposed mechanism used "experience-based + random" to forward a request in the network. With the addition of the reservation algorithm, more available matching resources can be found by using TTL value in the user's request message. The deadline for resource discovery is determined by TTL value. In this mechanism, only one resource is automatically allocated for any request if multiple available resources are found on forwarded path of resource discovery, resulting in no need to ask user to manually select the resource from a large list of available matching resources. The performance of the proposed algorithms is evaluated comparing with FFFS algorithm and the experimental results show that the percentages of request that can be supported by both algorithms are not different. However, it can improve the performance of either resource utilization or turnaround time, depending on how to select the resource. But this mechanism needs more hops to obtain a better solution than FFFS approach and in the high request traffic environment, the performance is not much different from FFFS algorithm (Tangpongprasit et al., 2005). Also the authors did not

Table 2
Popular decentralized mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
Iamnitchi and Foster	Offers a flat, decentralized, self-configuring architecture and is based on FFFS	Provides high load balancing	the first resource is allocated to requester without more searching, does not have high scalability, are inefficient in response time, cannot guarantee to find requested resources
Zhu et al.	Resources are discovered according to their attributes rather than their identifiers via DHT	Improves the performance of Grid resource discovery, high load balancing and scalability	The maintenance overhead was increased, constriction on relations between different resources was not considered
Tangpongprasit et al.	It is based on TTL reservation and unicast request forwarding and used "experience-based+random" to forward a request.	Improves the performance of resource utilization more than FFFS, high load balancing and scalability	The mechanism needs more hops to obtain a better solution in comparison to FFFS and it suffers from false-positive error
Li and Liu	Grid nodes are clustered based on keyword combinations and the mechanism realizes knowledge sharing and knowledge integration in knowledge Grid	The efficiency, load balancing, scalability and the success rate of resource discovery are improved	The efficiency and effectiveness of the proposed method were not verified through simulation experiments
Brocco et al.	An ant colony algorithm which is used with the participation of six types of ant	Improves hit rate, load balancing and scalability, minimizes degree of each node and bandwidth requirements	Suffers from high overhead and response time
Fouad et al.	Is based on UDDI but is not focusing on the internal structure of it, search model is based on sending DNS queries	Provides low cost, high scalability and load balancing	Multiple clients and multiple server nodes were not considered
Kocak and Lacks	Uses the programmable networking hardware and discovering Grid resource in five phases: subscription, advertisement, transaction, sign-off, and retirement	Offers high scalability, security and load balancing, gives autonomy to the task packets	Average hop count, average hit ratio and control message overhead can be improved

consider some important issues in Grid systems such as: the effect of network size, cost and budget of job execution, advanced reservation, and parallel job execution. Also, the proposed mechanism cannot guarantee to find requested resources even if they exist in the network (false-positive error).

Li and Liu (2007) presented a new decentralized resource discovery based on keyword combinations and node clusters. The main contribution of this mechanism is to explore how to support user's knowledge requests submitted in form of multi-keywords. In this mechanism, hot keyword combinations are formed based on user's knowledge requests and user's knowledge requests will be transmitted to those clusters that have high correlations with the requests (Li and Liu, 2007). Because knowledge request is transmitted to several target clusters, the efficiency and the success rate of resource discovery are improved. Also this mechanism can support knowledge retrieval based on multi-keywords, which fits to realize knowledge sharing and knowledge integration in knowledge Grid.

Brocco et al. (2010) proposed a distributed approach to the problem of resource discovery in a self-structured Grid. The authors employ an ant colony algorithm to form and support a P2P overlay network of all the Grid nodes with a minimal number of links and a restricted network size. Six types of ants participate in the ant colony overlay formation and optimization process: discovery, construction-link, optimization-link, unlink, update neighbors and ping ants (Brocco et al., 2010). Discovery ants perform random walks across the network and store collected information about the visited nodes in the alpha-table of their originating node. Construction-link ants are originated from new peers that want to enter the network. If a destination node is at its maximum capacity, the ant is forwarded to survey its neighboring nodes and once a suitable node is found, it is added to the neighborhood set of the originating node. The destination node is added to the neighborhood set of the originating node. Optimization-link ants form connections between nodes only if the connection meets the optimization requirements defined by a connection rule. Ants remove any existing links between two nodes if a disconnection rule applies or a node leaves the network. The proactive caching is accomplished similar to gossiping algorithm by periodically gathering information about resources similar to the requesting node and is stores it in a local cache. Furthermore, with respect to the overlay network, average path length is constrained and the degree of each node is minimized along with the number of unnecessary links due to caching (Brocco et al., 2010). But this mechanism suffers from additional overhead which is created by managing the network architecture. This overhead affects the response time.

Fouad et al. (2011) presented scalable Grid resource discovery through a distributed search. The proposed mechanism is similar to the Benson et al. (2006), but they did not focus on the internal structure of UDDIs for discovering Grid services. The proposed model includes the application layer which provides a web interface for the user and the collective layer which is a web service to discover resources. The resource discovery model contains the metadata and resource finder web services to provide a scalable solution for information administrative requirements when the Grid system expands over the Internet. The authors addressed two separate problems: providing a scalable solution and examining the scalability of the solution. Distributed search in this model is based on sending Domain Name Service (DNS) queries to the repositories with similar zone files that are distributed in the Internet. It first searches for resources from the regional repository, and if the required resource is not found, then another repository outside the regional domain will be contacted. The authors claim that if the proposed model were implemented on the Internet with thousands of Grid resources and users, the cost would be negligible, which means the model would be scalable (Fouad et al., 2011). But the architectures of proposed models are implemented with one client and one server on the same machine as the Grid simulator; therefore, multiple clients and multiple server nodes were not considered. Also, details of the whole model of repositories were not discussed.

Kocak and Lacks (2012) presented a distributed resource discovery protocol in Grid by using the programmable networking hardware to enhance scalability and security concerns. The proposed protocol eliminates some of the issues related to the common central resource broker scheme such as being a single point of failure, maintaining a fresh view of the overall Grid and giving autonomy to the task packets. The proposed protocol discovers Grid resource in five phases: subscription, advertisement, transaction, sign-off, and

retirement. During subscription, the resource provider subscribes as a member of the Grid network. The advertisement phase allows the Grid to know about the available resource. The actual transaction occurs in the transaction phase and the trust factors are logged at the end of this phase. The sign-off phase occurs if the resource is unavailable for a period of time. The retirement phase is when the resource provider no longer wants to be a member of the Grid network or if the resource is forced off of the network because its trustworthiness rating is poor. The authors have shown that the discovery process does not appear to be impacted by the network size, memory consumption, score deviation, or number of hops each message travels (Kocak and Lacks, 2012). A simulation framework as a scalable simulator is developed to model the distributed Grid environment and allows many Grid scenarios to be created, tested and validated. But the proposed mechanism was not compared with other mechanisms. Average hop count, average hit ratio and control message overhead can be improved by LARD mechanism in Akbari Torkestani (2012).

3.2.3. Summary of decentralized mechanisms

Decentralized mechanism allows Grid client devices and Grid resources to dynamically join or leave the Grid by connecting or disconnecting to one or more resource routers. It also supports resource reservation and QoS through the computational economy services. Furthermore these mechanisms provide high scalability, fault tolerance, load balancing and avoid single point of failure problem. But most of these mechanisms need periodic update of resource information therefore they did not support dynamic attribute query. Also, resource discovery is slow in the network and more query traffic is occurred. Table 2 summarizes the main properties of popular decentralized mechanisms.

3.3. Peer to peer mechanisms

In this section, we first describe the Peer to Peer mechanisms of resource discovery and their basic properties. Then, several most popular Peer to Peer mechanisms are discussed in four main categories. Finally, these mechanisms are compared and summarized in Section 3.3.3.

3.3.1. Overview of peer to peer mechanisms

Centralized and decentralized mechanisms of resource discovery in Grid environments cannot act well in dynamic and large-scale distributed environments. The number of queries in such environments makes a client–server approach ineffective. Therefore, large-scale Grid systems should be implemented based on efficient resource discovery mechanisms which cover all mentioned issues. Peer to Peer (P2P) overlay technology has emerged as a popular way to share data across a large peer population and offers several advantages over the centralized approaches (Hawa et al., 2013; Trunfio et al., 2007). P2P systems consist of a dynamically changing set of nodes with symmetric roles connected via the Internet (Krauter et al., 2002). P2P systems are a class of self-organizing systems that takes advantage of distributed resources storage, processing, information, and human presence (Singh, 2001). In P2P systems peers have equal roles and capabilities to exchange information and to provide services (Tan et al., 2012), as opposed to the traditional client–server model. P2P computing as a popular type of P2P systems is a distributed computing paradigm that uses large numbers of autonomous hosts as a platform for executing applications (Castellà et al., in press). P2P Grid as a important type of P2P computing systems exploits the synergy between the Grid system and P2P network to efficiently manage the Grid resources and services in large-scale distributed environments (Akbari Torkestani, 2012). It merges the positive features from both P2P system and Grid, in particular, in-and-out flexibility and fast search mechanisms. The ultimate goal of building P2P Grid is to integrate the P2P, Grid, and web services (Marín Pérez et al., 2011). P2P Grid extends the resource management of Grid, and employs P2P infrastructure to quickly locate the required resources. P2P Grid is a highly dynamic and scalable environment in which the peers enter, depart, and rejoin the system frequently and unpredictably (Akbari Torkestani, 2012). In the next sub-section, most popular resource discovery mechanisms in P2P systems (such as Kademlia, DC++, Napster, KaZaA, Gnutella, JXTA, Chord and CAN) and P2P Grid (such as Akbari Torkestani, 2012; Andrzejak and Xu, 2002; Cai et al., 2004; Deng et al., 2009; Iamnitchi et al., 2002; Moreno Marzolla et al., 2007; Mastroianni et al., 2005, 2008; Moreno-Vozmediano, 2009; Puppin et al., 2005; Schmidt and Parashar, 2003; Ali and Ahmed, 2012) in 4 main categories: unstructured, structured, super to peer and hybrid are discussed.

P2P systems almost mixed up with decentralized systems whereas they can be centralized or decentralized (Groeper et al., 2009). In centralized systems an indexed server is used to maintain a database of its users at any time as well as the content shared. The database is updated whenever a peer logs on to the network. Query requests from peers are sent to the index server and the database is queried. If matches are found, the server returns the result to the initiator with information of the nodes. Transfer can then be initiated with this information. Examples of application developed using this architecture is Napster[4] and Bittorrent.[5] In decentralized systems instead of central servers, each peer acts as an index server, searches and holds its local resources. It also acts as something like a router, relaying queries between peers. A node will send a query message to the peers it is directly connected to. If there is a match in their list of resources, the query will be forwarded to the peers they are connected to on the network. This process continues across the whole network. However, there would be a lot of network chatter due to the amount of being querying done. Examples of application developed using this architecture are Content Addressable Network (Ratnasamy et al., 2001), Chord (Stoica et al., 2001) and Gnutella.[6] Some other systems such as (Rowstron and Druschel, 2001), KaZaA,[7] Gnutella2[8] and Morphus[9] are partially decentralized P2P systems. In these systems peers can have different roles. Some peers act as local central indexes for files shared by local peers. These special peers are called “supernodes” or “superpeers.”

Up to now many P2P systems have been introduced and adopted to Grid environments. Here, we divided these mechanisms into four main categories: unstructured P2P mechanisms, structured P2P mechanisms, hybrid P2P mechanisms and super to peer mechanisms which are described as follow.

3.3.2. Unstructured peer to peer mechanisms

In this section, first, the unstructured P2P mechanisms of resource discovery and their basic properties are described. Then, several popular unstructured P2P mechanisms are discussed. Finally, the discussed mechanisms are compared and summarized.

In unstructured systems each peer is randomly connected to a fixed number of peers and there is no information about the location of resources (Trunfio et al., 2007). In these systems peers maintain fixed number of connections with their neighbors; the required overlay network is built in this way (Shah, 2007). The

[4] www.napster.com.
[5] www.bittorrent.org
[6] www.gnutellaforums.com
[7] www.kazaa.com
[8] www.gnutella2.com
[9] www.morphus.com

Table 3
Popular unstructured P2P mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
Napster	It is the first commercial P2P system to share mp3-files	Offers simple client–server architecture	Contains bottlenecks and single point of failure, is not easy to scale and is specialized for a single functionality
Iamnitchi et al.	Divides the discovery process into four parts: membership protocol, overlay construction, preprocessing, and request processing	Supports attribute-based search and is not dependence on central control	Does not scale well, the search results are not deterministic and cannot guarantee to find the desired resource
Moreno Marzolla et al.	Describes a routing strategy based on bit-vector, applies bit-vector indices to two different peer topologies, and uses natural partitioning/distribution	Update and query propagation algorithms are efficient, and the declaration of multi-attribute range queries is simplified	Queries might not be able to retrieve all matching resources and IS carries very different computation and traffic loads
LARD (Torkestani)	Use learning automata and The resource queries are forwarded through the shortest paths	Relieve the global flooding Effect on the network performance, supports the multi-attribute range queries, decrease the hop count and message overhead, and increases hit ratio	Cannot guarantee the number of hops taken to deliver the services to requester

discussions and overview of popular unstructured P2P mechanism of resource discovery and their main properties are provided as follow:

In 1999 the first commercial P2P system was introduced to share mp3-files called Napster. Napster has been probably the most revolutionary and unprecedented example of digital music distribution over packet switched networks. This system comprised a central server which stored the index of all resources (files) shared by the peers. To locate a resource a user queried the central server using the name of the resource and received as a result the IP address of a peer containing the resource. Then a direct connection was established between the requesting peer and the peer containing the resource being downloaded. It is not easy to scale the central index server used in Napster and it has 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 (Trunfio et al., 2007). But it is not a pure P2P system because its database is centralized. Also, Napster is only specialized for a single functionality (MP3-Files) and had the problems of canalized mechanisms.

Gnutella was the second major P2P system after the Napster. It was constructed to be decentralized and followed the classical concept of an unstructured P2P system. Its overlay maintenance messages include *ping*, *pong* and *bye*, where *pings* are used to discover hosts on the network, *pongs* are replies to *pings* and contain information about the responding peer and other peers it knows about, and *byes* are optional messages that inform the upcoming closing of a connection. For resource discovery, a flooding strategy is used, where a query is propagated to all neighbors within a certain number of hops (Qiao and Bustamante, 2006). Therefore, the number of query packets typically increases exponentially and causing huge overhead. Additionally, due to frequent peers disconnects, this network was never stable. The cost of the search increases exponentially depending on the number of searched users. When the network grows large enough, it gets saturated and often caused enormous delays. But it will be efficient and scalable if it generates less number of redundant queries. Several improvements have been suggested to make Gnutella scalable, one of the suggested mechanisms was Gnutella2 that employs super-node architecture which is discussed in Section 3.3.4.

Iamnitchi et al. (2002) proposed a fully decentralized approach to resource location in Grid environments based on flat P2P networks. The architecture partitions the resource discovery solution into four components: membership protocol, overlay construction, preprocessing, and request processing. Every participant in a VO publishes information on one or more local servers. A node may provide information about one resource or multiple resources. Users send their requests to a known node; then if this node has the requested query it responds with a matching resource description; otherwise it forwards the requests to another node. Intermediate nodes forward a request until its TTL expires or matching resources are found (Iamnitchi et al., 2002). This mechanism supports attribute-based search and is not dependent on central control. However, it does not scale well because of the large volume of query messages generated by flooding. The search results are not deterministic and this approach cannot guarantee to find the desired resource even if it exists. Also in order to avoid flooding of the complete network, the number of hops on the forwarding path is bounded by the TTL (Akbari Torkestani, 2012)

Moreno Marzolla et al. (2007) proposed P2P systems to discover resources in a dynamic Grid in which the peers hold a set of local resources, each resource is described by a set of associated attributes. Attribute values vary over time, and this makes most of the existing P2P approaches not adequate. Users can locate resources by performing range queries over the set of all attributes. The authors described a routing strategy based on bit-vector RI (Routing Index), which can be used to route queries towards nodes of the system where matches are likely to be found. Moreover, the bit-vector indices can effectively be updated when attribute values change. They applied bit-vector indices to two different peer topologies. The first one is a simple topology based on a single tree, where peers are connected via a tree-shaped overlay network. Second one is a more relaxed network topology based on a forest of trees: in this network multiple groups of nodes exists, internally connected as a tree, while the inter-group connections are arbitrary. One of the main features of the proposed mechanism is simplifying the declaration of multi-attribute range queries. Many mechanisms have resolved multi-attribute queries by means of a separate DHT for each attribute type. In these cases, the final result is lists of matching resources. Moreover, even if the average complexity of locating a single item by using a DHT is usually logarithmic in the size of the network, the complexity may become linear in the case of queries asking for very large attribute ranges. This is because many DHT nodes, each responsible for a small range of attribute values, must be contacted sequentially. Unlike the DHT-based networks, the proposed approach is based on a natural partitioning/distribution of the global index to the various peers of the network, simply entailed

by the resources that have been assigned to each peer node. Each node can thus compute its local index on the basis of its own resources, and all the attributes associated with them. Finally, every node is capable of locally resolving all the subqueries on every type of attribute. This network supports frequent updates of the attribute values of resources, without the need to broadcast changes to all the nodes. They proposed a second solution based on a forest of trees. This solution not only preserved features of the simplicity, but also introduced a hierarchical P2P network that is easier to maintain and manage. RI information can also be maintained easily in the forest-based approach, with the drawback that queries might not be able to retrieve all matching resources, thus obtaining a recall that is less that 100% (Moreno Marzolla et al., 2007). However, they showed that, under reasonable conditions, the system can still achieve a good recall, while the network topology ensures a limited radius of query propagation. This happens if the inter-group connections of this hierarchical network topology are modeled as a scale-free network and the average group size is large enough. The simulation results were supported by analytical evaluations in order to assess the performance of the query and update routing algorithms for the single tree and forest scenarios. As performance measures they considered the number of hops of messages, precision and recall of queries, and number of nodes receiving a message. In this mechanism fault-tolerance is limited by the presence of a bottleneck at the tree root; a significant amount of memory space must be reserved in Index Services to maintain information about a large number of resources causing limitation of the scalability of the Grid; Index Services belonging to different levels must carry very different computation and traffic loads leading to challenging problems concerning load imbalance; and the hierarchical organization can hinder the autonomous administration of different organizations (Mastroianni et al., 2008). The authors showed that the update and query propagation algorithms are efficient since they do not propagate messages over the whole network.

LARD (Learning Automata-based Resource Discovery) is another unstructured P2P mechanism which is proposed by Akbari Torkestani (2012). LARD is a decentralized resource discovery algorithm for large-scale unstructured P2P Grids in order to solve the problems of the previous methods. In the proposed resource discovery algorithm, he has used learning automata in order to find the shortest path which connects (the path with the minimum hop count) the user to the peer which provides the requested resource. In the proposed method, each peer chooses a communication link to route the resource provider, so that this link is selected by the automaton randomly. If the selected route at each stage is shorter than the average length of the routes selected so far, algorithm rewards the selected route, otherwise it is penalized. Thus, as the proposed algorithm proceeds, algorithm converges to the route having the minimum expected length. The proposed algorithm supports the high dynamicity of the scalable P2PGrids where the peers frequently and unpredictably joins, leaves, and rejoin the system (Akbari Torkestani, 2012). This algorithm relieves the global flooding effect on the network performance and supports the multi-attribute range queries too. Experimental results showed that the proposed algorithm have better performance than Deng et al. (2009) and Kocak and Lacks (2012) in terms of average hop count, average hit ratio and control message overhead in all small, medium, and large scale Grids. But it cannot guarantee the number of hops taken to deliver the service to requester.

In this section an overview of popular unstructured P2P mechanisms and their important features are provided. As discussed, these mechanisms do not fully support the dynamic and multi-attribute range queries and suffer from the network-wide broadcast storm problem. Most of the methods in this category do not scale well because each individual query generates a large amount of traffic and the network quickly becomes overwhelmed by the messages (Deng et al., 2009). Also, the system suffers from overhead of flooding and query may not be answered. Considering the nature of the unstructured P2P mechanisms, in most cases, because of the common routing mechanisms, the complexity of the algorithms is around $O(N^2)$, which makes the approaches unsalable. Moreover, in some cases time complexities have higher order of growth than the scale of the network. Moreover queries are not lost in the network and propagation of the queries continues until a TTL value is reached. In many mechanisms in this category it is possible the algorithm may return unsuccessful results even if the searched resources exist and are available on the Grid because of the TTL limit are reached (false-positive error). Otherwise, when TTL is set to a higher value, asymptotic increase in the messages negatively affects the bandwidth and runtime of the algorithms (Hameurlain et al., 2010). But unstructured P2P mechanisms are reliable in terms of query correctness and single point of failure, and can tolerate node dynamicity. Also most of unstructured P2P mechanisms support range, multi-attribute and dynamic-attribute queries easily. Table 3 summarizes the main properties of popular unstructured P2P mechanisms.

3.3.3. Structured peer to peer mechanisms

In this section, the structured P2P mechanisms of resource discovery and their basic properties are described. Then, seven most popular structured P2P mechanisms are discussed. Finally, the discussed structured P2P mechanisms are compared and summarized.

Structured P2P systems employ an inflexible structure to interconnect the peers and organize the resource indices (Trunfio et al., 2007). These systems are prepared with a distributed indexing service which is based on hashing, and is known as DHT. Peers and resources are mapped, usually through the same hash function, to a key space (Trunfio et al., 2007). However, they do not support direct keyword searches which constitute the core of queries in real P2P systems. In this section seven most important mechanisms to discover resources in structured P2P systems are discussed as follow:

The fist structured P2P system is Chord which is a scalable protocol for lookup in a dynamic P2P system with frequent node arrivals and departures that was introduced in 2001 (Stoica et al., 2001). In Chord, both peers and resources are mapped through the same hash function to an m-bit key space (Stoica et al., 2001). The peers in Chord are organized in a one-dimensional circle according to their keys (Trunfio et al., 2007). Each peer stores the index of all resources whose keys fall in the range between the key of its predecessor and its own key (Stoica et al., 2001; Trunfio et al., 2007) and the lookup process emulates the binary search. While DNS based approaches such as (Fouad et al., 2011) rely on a set of special root servers, Chord requires no special servers, while DNS names are structured to reflect administrative boundaries, Chord imposes no naming structure, and while DNS is specialized to the task of finding named hosts or services, Chord can also be used to find data objects that are not tied to particular machines. Chord presents scalable single key-based registration and lookup service for decentralized resources. Since each peer is responsible for an equal number of keys with high probability, thus load balancing is achieved (Dabek et al., 2001). But Chord cannot support range queries and multi-attribute-based lookups and suffer from lookup latency. Hybrid-Chord by Flocchini et al. (2005) enhances the Chord performance and robustness by introducing some redundancy in the system via laying multiple chord rings on top of each other and using multiple successor lists of constant size. Chord-based DNS (Cox et al., 2002) and Cooperative mirroring/

Cooperative File System (Dabek et al., 2001) are other examples of the systems that used Chord.

The Content Addressable Network (CAN) is another structured P2P system that was introduced by Ratnasamy et al. (2001). Unlike Chord, it offers a clearly defined structure which can be secured with relatively low effort (Hof et al., 2007). CAN uses greedy routing strategy where a message is routed to the neighbor of the current node that is situated closer to the required location (Meshkova et al., 2008). The basic operations of the CAN are insertion, lookup and deletion. The proposed system tries to limit the number of each peer's neighbors, regardless of the size of the network or the key space (Ratnasamy et al., 2001; Trunfio et al., 2007). The peers in CAN are organized in a d-dimensional torus. Each peer is connected to its next and previous peer in each dimension (Ratnasamy et al., 2001). 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. An extension of CAN employs more than one hash function in order to support replication and thus to reduce lookup cost and to provide fault tolerance in the case of unpredictable peer departures (Ratnasamy et al., 2001; Trunfio et al., 2007). The CAN is highly scalable, robust, fault-tolerant and self-organizing (Meshkova et al., 2008); but does not support range queries completely; furthermore, it does not provide any additional mechanism to increase data availability, and the data is lost when some nodes are crashed. D2B (Fraigniaud and Gauron, 2006) and SCAN (Hof et al., 2007) are examples of the CAN based systems.

Andrzejak and Xu (2002) extended the CAN-based DHT-system into an indexing infrastructure which allows querying of ranges and supports efficient handling of dynamic data by using one particular Space Filling Curve (SFC), Hilbert curve, which is a continuous mapping from a d-dimensional space to a 1-dimensional space f: $N^d \rightarrow N$. Expressway routing in CAN is used to further cut down costs of searching and updating (Shen, 2009). The proposed mechanism used three simple strategies for propagating range-query requests, and strategies to minimize the communication overhead during the attribute updates (Andrzejak and Xu, 2002). The effectiveness of these strategies is evaluated through simulations and the authors show that the proposed mechanism is effective in meeting the goals of scalability, availability and communication-efficiency. The work provides foundation for a self-organizing and scalable implementation of a Grid information infrastructure as Grid index information service, which provides a coherent image of distributed Grid resources and allows searching for specific resources (Shen, 2009). This mechanism complements MDS-2 (Czajkowski et al., 2001) by adding self-organization, fault tolerance and an ability to efficiently handle dynamic attributes, such as server processing capacity. But the mechanism needs higher maintenance costs, since it is built on top of a DHT and therefore requires one additional mapping step and it also supports large multidimensional range queries and in a single lookup. Furthermore, in case of adding new resource attributes, the additional dimensions are needed, therefore the number of neighbors and management overheads are increased. In addition, the dedicated subsets of servers are responsible for specific ranges of an attribute which causes the algorithm to support range attribute queries and since different types of attribute are mapped to a distinct DHT, the multi-attribute queries are supported.

Kademlia is another type of structured P2P systems which is a distributed hash table for decentralized P2Pcomputer networks based on the XOR metric which is designed by Maymounkov and Mazières (2002). Unlike Chord, CAN, or Pastry (Rowstron and Druschel, 2001), Kademlia uses Tree-based routing. In this system each peer is mapped to a 160-bit key through a hash function. A peer usually stores data items whose key values are close to its peer ID. 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". 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 the lookups to refresh a bucket's contact is thus performed rarely, usually by new peers, during their bootstrap phase. Also, there is no need for a departing peer to leave gracefully, since stale bucket entries are purged. Bittorrent DHT Protocol, Khashmir[10] are the examples of Kademlia's implementation. Kademlia contacts only $O(\log(n))$ nodes during the search out of a total of n nodes in the system (Maymounkov and Mazières, 2002). Furthermore, Kademlia uses parallel asynchronous queries to avoid timeout delays of the failed nodes. Moreover, Kademlia minimizes the number of configuration message nodes. Kademlia is the first P2P system that: (1) combines provable consistency and performance, latency minimization routing, and symmetric, unidirectional topology; (2) introduces a concurrency parameter, which lets people trade a constant factor in bandwidth for asynchronous lowest-latency hop selection and delay-free fault recovery; and (3) exploits the fact that node failure are inversely related to uptime (Maymounkov and Mazières, 2002). But Kademlia uses a very complex process of peer discovery, which is prone to implementation errors. Also, it involves a great deal of additional overhead caused by bucket refreshes and so on. Therefore many works such as Binzenhöfer and Schnabel (2007) have improved the performance and robustness of Kademlia.

Schmidt and Parashar (2003) proposed a decentralized single-dimensional DHT-based information discovery technique supporting multi-attribute queries. In this mechanism, each resource has multiple attributes so that it is mapped into the node where its ID is obtained by interleaving the binary representation of the attributes value. The proposed architecture is a DHT, similar to CAN (Ratnasamy et al., 2001) and Chord (Stoica et al., 2001) but the proposed mechanism used different methods to map data elements to the index space. In previous mechanism, the mapping is done by using consistent hashing; therefore data elements are randomly distributed across peers without any notion of locality. This mechanism preserve locality while mapping the data elements to the index space. In this mechanism, all data elements are described using a sequence of keywords for resource discovery in computational Grids. These keywords form a multidimensional keyword space (Schmidt and Parashar, 2003). If the keywords of two data elements are lexicographically close or they have common keywords they are "local". The mechanism uses the Hilbert SFC (Andrzejak and Xu, 2002) for the mapping, and Chord (Stoica et al., 2001) for the overlay network topology. The experiment results demonstrated the scalability of the system, and showed that only a fraction of the total nodes in the system typically process a query and this fraction is almost the same as the nodes that store data elements matching the query. The results also showed the ability of the mapping to preserve keyword locality and the effectiveness of the load balancing algorithms. But some issues such as hot-spots, fault-tolerance, security and resistance to attacks, and maintenance of geographical locality in the overlay network were not considered. Also some extra joining and migration overhead is appeared.

Cai et al. (2004) proposed MAAN which handle multi-attribute range queries by extending Chord (Stoica et al., 2001) with locality preserving hashing and a recursive multidimensional query resolution mechanism. For attributes with numerical values, MAAN uses locality preserving hashing functions to assign each attribute value an identifier in the m-bit space, and then maps the value information to Chord. Also it distributes resources to all nodes

[10] www.khashmir.sourceforge.net

Table 4
Popular structured P2P mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
Chord (Stoica et al.)	Both peers and resources are mapped through the same hash function, each peer is responsible for an equal number of keys with high probability	Provides load balancing and high scalability	Does not support range and multi-attribute queries and suffers from lookup latency
CAN (Ratnasamy et al.)	Uses greedy routing strategy, limits the number of each peer's neighbors and basic operations are insertion, lookup and deletion	Provides high scalability, robustness, fault-tolerant and self-organizing	Does not support range queries, and data availability is low
Andrzejak and Xu	Extends the CAN-based DHT-system by using SFC	Provides high scalability, availability, communication-efficiency, and supports multi-and dynamic attributes queries	Needs higher maintenance costs, and the additional dimensions are needed, management overheads are increased
Kademlia (Maymounkov and Mazières)	Uses Tree-based routing, XOR metric for distance between peers and single routing algorithm from start to finish	Contacts only $O(\log(n))$ nodes during the searching process, minimizes the configuration messages nodes, and avoid timeout delays of the failed nodes	Uses a very complex process of peer discovery, is prone to implementation errors and additional overhead
Schmidt and Parashar	Uses Hilbert SFC mapping	Provides high scalability, load balancing and query processing is done by small number of nodes	Suffers from extra joining and migration overhead
MAAN (Cai et al.)	Uses locality preserving hashing and a recursive multidimensional query resolution	Supports multi-attribute and range queries	The attribute schema of resources has to be fixed and known in advance
ACO (Deng et al.)	Represents a query message by ant	Improves searching efficiency, increasing hit ratio	Supports multi-attribute range query

uniformly and achieves good load balancing among node. MAAN can support multi-attribute range queries well, but, the attribute schema of resources has to be fixed and known in advance with MAAN. MAAN supports multi-attribute queries by constructing multiple DHTs for each attribute and supports range queries by using hashing functions. But when the range of queries is very large, flooding the query to the whole network can actually be more efficient than routing it to nodes one by one as MAAN does (Cai et al., 2004). Also since DHT is generated once and is updated at discrete intervals, the dynamic-attribute queries are not supported.

An ACO (Ant Colony Optimization) based resource discovery algorithm for large-scale peer to peer Grid systems is proposed by Deng et al. (2009). In this mechanism each ant represents a query message. At first, the ants walk randomly from nest to nest to locate resources. If the ants find the required resources, they will take the same path to return to their original nest and update the routing information on the path in terms of their memory. The other ants which are looking for the same resources will travel in the system according to the routing information. Therefore, most of the ants are most likely to choose the shortest path to travel in the system. This method avoids a large-scale flat flooding of the unstructured method by sending the packets along the routes that are frequently traveled by the ants while saving the network resource consumption. In addition, the searching efficiency can also be improved by employing multiple ants which can work in parallel. Compared with the structured method, the simulation demonstrated that this method takes longer network distance than the hash based method. However, it has much higher hit ratio (80.1% against 42%) (Deng et al., 2009).The ants in the ACO method can carry a large amount of information in their memory when it is required. Since, multiple user requirements stored in memory, the mechanism supports multi-attribute range query. This feature is very important for a Grid system, because the Grid users should be able to locate resources with multiple requirements.

In these mechanisms the information about the resources is stored in a node specified by a key. These mechanisms can support resource diversity and provides scalability, robustness, and self-organization. Since these algorithms use topological structures, time and message complexities of the algorithms are around $O(\log N)$ (Hameurlain et al., 2010). In many algorithms, all resource nodes get involved in the query processing, which means that, theoretically, all nodes will have the same load. This eliminates the bottlenecks in the system and ensures the scalability of the structured P2P approach. But, when the information is changed, the change should be distributed through the network; therefore, a considerable amount of network traffic is created. On the other hand, its propagation of changes is slow, which makes such a system inappropriate to store rapidly changing information. So, structured peer to peer network suffers from the network-wide broadcast storm problem and need strong self-organizing mechanisms in order to maintain their fixed structure. Also, these methods do not fully support the multi, range and dynamic attribute queries. However, they are reliable in terms of query correctness and single point of failure. Table 4 summarizes the main properties of popular structured P2P mechanisms.

Next section reviews the super to peer mechanisms and provides some popular super to peer mechanisms and their basic attributes.

3.3.4. *Super to peer mechanisms*

In this section, a basic description and overview of super to peer mechanisms are provided. Then some of popular super to peer mechanisms of resource discovery are discussed and reviewed. Finally, the provided mechanisms are compared and summarized.

The super to peer mechanism is a novel approach that facilitates the convergence of P2P models and Grid environments, since a super-peer serves a single Virtual Organization (VO) in a Grid and at the same time connects to other super-peers to form a peer network at a higher level (Mastroianni et al., 2005). In a super-peer network, all peers can be classified as peers and super peers (Tan et al., 2012). A super peer acting as a server to connect many client peers, and is responsible for tasks such as searching and routing (Tan et al., 2012). These mechanisms have been originally proposed to achieve a balance between the inherent efficiency of centralized search, and the autonomy, load balancing

Table 5
Popular super to peer mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages1
KaZaA	Two types of nodes exist, super nodes and leaf nodes; the leaf node reports its resource indexes to the super node.	Provides high scalability.	Does not support complex queries; and communication among super peers is not well organized.
Gnutella2	Has two types of nodes: leaves and hubs; leaf sends a search request to a hub.	Offers scalability, reducing the traffic.	Suffers from high complexity; vulnerability against attack.
Mastroianni et al.	Includes two types of peer: super peer and regular peer; a regular peer sends its request to its local super-peer; Super peer returns the response.	Provides autonomy, load balancing and fault-tolerant features.	Suffers from single point of failure in each cluster and false-positive errors. Dynamic attribute queries are not supported.
Puppin et al.	Includes two main components: agent and aggregator.	Provides integration with any Globus-based Grid; scalability; low traffic and supports range and multi-attribute queries	Suffers from single point of failure on each cluster and false-positive errors. Dynamic attribute queries are not supported.
Ali and Ahmed	It is constructed by two layers called HyperSN which is connected using ring topology and CHyperSN which is a set of HyperSN.	Provides high scalability, reliability and supports range and multi-attribute queries.	Suffers from high traffic and single point of failure in each SN.

and fault-tolerant features offered by distributed search (Beverly Yang and Garcia-Molina, 2003; Trunfio et al., 2007).

KaZaA is a P2P system that has adopted the super-peer model in its design, which was introduced in March 2001. KaZaA is a peer-to-peer resource (file) sharing program used by many millions of people to share resources without any servers. More correctly KaZaA is the name of one of the programs used on this particular machine to connect to a peer-to-peer network (Sanderson, 2006). In KaZaA two types of nodes exist, super nodes and leaf nodes. An overlay network is formed among the super nodes, each of which carries a set of leaf nodes. The leaf node reports its resource indexes to the super node when it joins a super node. When a node looks up a resource, it issues a request to its super node, which initiates a search process in the overlay network to locate the resource (Chen et al., 2008). KaZaa is more scalable than Napster and Gnutella because of its hierarchical architecture. But KaZaa cannot support complex queries as the queries are routed regardless of their content. Furthermore, the communication among super peers is not well organized and thus flooding or partial pooling is used, which is inefficient. In addition, it suffers from a large amount of false resources that users can encounter during their discovery KaZaa no longer offers a music service after August 2012.

As discussed before, Gnutella does not scale well in very large networks and suffers from drawback caused by flooding. Gnutella2 tries to solve the disadvantages of Gnutella which was released in 2002. Gnutella2 has two types of nodes: leaves (normal peer) and hubs (cluster-heads or super-peers). Each leaf, maintains one or two connections to hubs. Cluster-heads index resources of hundreds of peers by means of a Query Routing Table. The connected hubs also exchange hashes of keywords describing the resources that their leaves provide. During the search a leaf (peer) sends a search request to a hub. If it answers the peer downloads the file directly from the peer that hosts the resource. If the search is not successful, the request is forwarded by the current hub to another hub. Its address is taken from the routing tables of the super peers. The search stops when either the item is found or all known hubs are searched or a predefined search limit is reached. This approach considerably reduces the traffic in the network and makes the system much more scalable compared to original Gnutella. However, as a tradeoff, the complexity of Gnutella2 is higher than Gnutella and required additional network maintenance. And vulnerability of the system to DoS and other malicious attacks on the cluster-heads increases (Meshkova et al., 2008).

Mastroianni et al. (2005) adopted the super-peer model to design a P2P-based Grid information service. This model includes two types of peer: super peer and regular peer. Each super peer is related to a number of local regular peers. Super peers are connected by an overlay peer to peer network. A regular peer sends its request to its local super-peer. Super peer returns the response, if it finds a local peer providing the requested service. Otherwise, it forwards the request to its neighboring super-peers (Mastroianni et al., 2005). This mechanism provides autonomy, load balancing and fault-tolerance features for resource discovery. But since a super-peer node acts as a centralized server for a number of regular peers, this mechanism suffers from single point of failures in each cluster. Moreover, when the scale of the requests increases, the super-peers may suffer from bottleneck, which may limit the scalability of this system. Also it suffers from false-positive errors, which are caused by usage of TTL parameters in flooding operations. Finally this mechanism cannot support dynamic-attribute queries because of the periodic updates of the resource information.

Puppin et al. (2005) proposed a super-peer based resource discovery scheme. In this mechanism, Grid nodes are divided into clusters, each having one or more super-peers. This model includes two main components: agent and aggregator. Each aggregator plays the role of a super- peer responsible for data collection, query processing and forwarding, and information indexing. A peer to peer network connects the neighboring super-peers. At each cluster, agent publishes the information of the provided resources. The Information System is built as a network of super-peers, which aggregate the data about resources within a virtual organization. Queries performed by any client are passed among the super-peers, using optimization algorithms such as the Hop-Count Routing Index. This system is based on Globus Toolkit and complies with the OGSA standard. It can be easily integrated with any Globus-based Grid. The authors used it for resource monitoring and discovery, but it could be used for file-sharing or other distributed applications. The authors tested the proposed mechanism using a small network, and they obtained that the system scaled effectively and the total traffic is reduced. However; the system suffers from single point of failure on each cluster and false-positive errors. Moreover, Periodic updates of resource information may require very frequent updates in highly dynamic networks and limits the dynamic attribute queries. But the range and multi-attribute queries are supported.

Ali and Ahmed (2012) proposed scalable framework for resource discovery in P2P based hypercube computational grid. The proposed framework is constructed by two layers called Hypercube Service Node (HyperSN) which is connected using ring topology and Circle Hypercube Service Node (CHyperSN) which is a set of HyperSN. SN is an ordinary node which has some better properties comparing with other nodes within organization i.e. nodes which have high availability, high CPU speed and high bandwidth connection. The proposed mechanism chooses the best node in term of reliability from ordinary nodes which use as index discovery service node, and mechanism to balance the load in the CHyperSN overlay.

Table 6
Popular hybrid P2P mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
DirectConnect (DC++)	Combines centralized and super to peer mechanism	Shares a minimum number of resources presenting a large diversity of data on the network	Has a bottleneck; is unable to handle all the network traffic of large-scale networks; scale poorly
JXTA (Waterhouse, 2001)	Peers are organized in peer groups, each peer group establishes a set of services	Is more suited for lightweight, flexible communication, has less administration costs	Broadcasting a request can reduce the performance
Papadakis et al.	The peers are divided in two categories (Superpeers and Peers), each Superpeers acts as a server	Provides global scalability, low lookup cost, reduces the number of duplicate messages	Suffers from slow query response time, large overhead
Moreno-Vozmediano	Combines peer-to-peer mechanisms and clustered solutions	Is scalable, low discovery delays and bandwidth consumption, and is adaptable to changing conditions	The node could forward the same query request several times, and the query requests may be forwarded to zones that were already visited

This mechanism provides the balanced load between SNs in the whole organization even in the environment with a highly skewed resource distribution. It also provides a preserving locality protocol based on a distance metric for the HyperRN overlay construction. This framework preserves administrative control and autonomy because routing between various administrative organizations is permitted according to the policies defined by the target organization. The proposed method is suitable for large scale distributed resource sharing systems with heterogeneous resources and different sharing policies. Also it is scalable in term of time, because it keeps the maximum number of steps required to resolve a range queries. Moreover it can support range and multi-attribute queries. This mechanism used flooding in Node Join phase; therefore it suffers from high traffic. Also the single point of failure exists in each SN.

These mechanisms achieve a balance between the inherited efficiency of centralized search, and the autonomy, load-balancing and fault-tolerant features offered by distributed search. The super-peer model can be advantageously adopted in large-scale Grids allowing a very efficient implementation of the information service. In super-peer mechanisms the communications take place only among super-peers, therefore it is most appropriate to ensure extensibility and scalability of the system (Mastroianni et al., 2005). But these mechanisms suffer from the difficult implementation and use of the rich resource description. Furthermore, due to the disjointed information source and the point of storage, the network load in dynamic environments is considerable. Also, the clustering procedure may become more complicated. In addition, the systems suffer from the bottlenecks when the number of request for the super-peer is very large. All of the algorithm in this category have message complexities $O(S^2)$ and time complexities $O(S)$ where S is the number of super-peers in the network (Hameurlain et al., 2010). Even these types of algorithms can be considered as more scalable than unstructured systems; super-peers may suffer from being bottlenecks in the system when the number of requests is large. Finally, single point of failure exists in each cluster. Table 5 provides a brief summary of super to peer mechanism and their main properties.

3.3.5. *Hybrid mechanisms*

In this section, first, we present a basic description and overview of hybrid mechanism, and then some of popular hybrid mechanisms of resource discovery are provided. Finally the provided mechanisms are compared and summarized.

Hybrid mechanisms have been proposed to overcome the drawbacks of mentioned P2P mechanisms by combining them while retaining each mechanism's benefits and advantages. These mechanisms benefit from the efficiency of each combining mechanism, while overcoming their inherited drawbacks. In the next section several hybrid mechanisms and their main features are discussed.

DirectConnect[11] is based on hybrid P2P architecture and associative clustering, while each interest group is being guided by a cluster head (hub) which was proposed by Jonathan Hess in November, 1999. Hubs are pieces of software that organize the life of each cluster, but do not participate in the resource exchange process which are located on central servers. All hubs are registered on the HubListServer, which then acts as a name service. Clients discover hubs by asking the HubListServer. A user can freely choose interesting group/cluster and can directly exchange resources or information with any other user in the same cluster in a P2P fashion. The DirectConnect protocol is a text-based protocol, in which commands and their information are sent in clear text, without encryption. As clients connect to a central source of distribution (the hub) of information, the hub is required to have a substantial amount of upload bandwidth available. DirectConnect requires a user to run a hub even on local area networks. The hub is used for address discovery, keyword searches and chat. Hubs facilitate communication between clients and give information about them while responding to resource discovery queries. However, in DirectConnect every peer shares a minimum number of resources to ensure that a large diversity of data is presented on the network, which may increase the probability of finding requested resources. But because of using central servers the hubs have a bottleneck and are unable to handle all the network traffic of large-scale networks. Also, DirectConnect scales poorly to larger networks.

JXTA (Juxtapose) is a distributed search system designed for hybrid P2P architecture in March 2001 by Sun Microsystems (Waterhouse, 2001). The JXTA is locally centralized, globally decentralized. Also JXTA's search algorithm is a hybrid mechanism using a DHT and a Random Walker. The JXTA platform is divided into three layers: Core Layer, Service Layer, and Application Layer and do its functionality with six XML-based protocols. Peers in a JXTA network are expected to interact through the services they offer/consume. Peers are organized in peer groups, where each peer group establishes a set of services. To describe a JXTA resource an XML based advertisement is used. Commonly, peer groups are used to organize peers offering services in a specific application domain (Waterhouse, 2001). The most important advantage of JXTA is their interoperability with any other digital device and their platform is independent. JXTA is more suited for lightweight, flexible communication. It has significantly less administration costs and is better suited for limited capability devices (Ashri, 2003). But broadcasting a request can reduce the

[11] www.dcplusplus.sourceforge.net

performance when weak nodes like modem links slow down the propagation of queries.

Papadakis et al. (2007) designed and implemented a hybrid P2P-based Grid resource discovery system which supports both static and dynamic information retrieval, and push and pull models. The search for static information is performed in a structured-like fashion, while dynamic information search is performed in an efficient unstructured-like fashion tailored to the DHT structure. In addition, this mechanism couples the structured topology with a broadcast method of unstructured systems to locate dynamic information. Thus, the proposed system is hybrid in more than one aspect. The mechanism combines the completely decentralized P2P paradigm with a limited degree of centrality to reduce the effort of providing a global view of the system resources. In proposed mechanism the peers divided into two categories (Superpeers and Peers) based on the level of service they can provide. Each Superpeers acts as a server for a number of regular Peers, while Superpeers connect to each other in a P2P fashion at a higher level. Unlike unstructured systems, the Superpeers were organized based on DHT-based system. This mechanism supports both a push and a pull approach of resource discovery which allows for a trade-off between message cost for resource discovery and staleness of provided information. Most participants act as normal Peers, while the high bandwidth participants act as Superpeers. Superpeers participate normally in the P2P overlay and also act on behalf of Peers, which participate in the system indirectly by connecting to Superpeers and causing improvement of the scalability of the system by exploiting the heterogeneity of participating nodes. In addition, Peers can provide their corresponding Superpeer with static information about the resources they manage. Thus, when a Superpeer receives a query, it can forward the query only to those Peers whose resources match the static criteria. The Peers will then reply with any local resource information that also matches the dynamic part of the query. While such approach is widely implemented by unstructured P2P systems, in this framework, the Superpeers are organized using Chord (Stoica et al., 2001), a well known DHT-based system. Not only the Chord structure can be used to quickly resolve queries based on static information; but also the structure of Chord allows distributing a query to all nodes in the overlay avoiding duplicate messages. Proposed mechanism allows distributing the query to as many nodes as it is required to locate the desired information, instead of flooding the entire network for every query. This means that the cost of the lookup is further reduced, depending on the amount of matching resources that exist in the system and the number of the results required by the user. But since three layers are used in this mechanism, a fast response to the queries is not provided if the queries do not answered at the lower layer. Furthermore the overhead of the system is noticeable.

Moreno-Vozmediano (2009) proposed a hybrid discovery mechanism, which combines the advantages of peer-to-peer mechanisms (high adaptability for changing conditions, and low management complexity) and the advantage of clustered solutions (high scalability). The author studied the existing resource and service discovery architectures, analyzing the main limitations of these systems (scalability, discovery delay, adaptation to changing conditions, etc.) and tries to overcome these limitations. It uses peer-to-peer communication, multicasting is restricted to the discovery zone, and queries are forwarded by peripheral nodes, avoiding flooding. This approach is based on the idea of zones, similar to the concept introduced by the Zone Routing Protocol (ZRP) (Haas and Pearlman, 2001). A discovery zone is defined for each Grid node individually, and is composed by all the neighbor nodes whose distance to the node in question does not exceed a certain number of hops (zone radius). This mechanism is scalable, exhibits low discovery delays and reduces bandwidth consumption, is adaptable to changing conditions, and does not require any management effort. But the node could forward the same query request several times, and it is possible that query requests may be forwarded to zones that were already visited.

In this section we discussed three important mechanisms and presented their advantages and disadvantages. A hybrid mechanism can be defined as a mixture of two mechanisms of which it uses their advantages. These mechanisms almost have high overhead but are reliable. The overview of mentioned mechanisms is summarized in Table 6.

3.3.6. *Summary of peer to peer mechanisms*

In this section we described most popular P2P mechanisms in four main categories. The discussions provide important features about P2P mechanisms. The most important advantages of these mechanism are first, these mechanisms are easy to set up; second, all the resources are shared by all the peers, unlike server based architecture where server shares all the resources; third, since central dependency is eliminated P2P mechanisms are more reliable therefore failure of one peer does not affect the functionality of other peers; fourth, there is no need for full-time system administration since every user is the administrator of his machine and can control their shared resources; fifth, the cost of building and maintaining these mechanisms are comparatively very low. The most important disadvantages of these mechanisms are first, since the whole system is decentralized, administration is difficult; second, security in these mechanisms is low; third, data recovery or backup is very difficult because each computer should have its own back-up system; fourth there is no guarantee about Quality of Service. Table 7 provides the summary and discussion about P2P mechanisms and their main properties.

Next section provides an overview and discussion about hierarchical mechanism of resource discovery in Grid environments as fourth discussed categories.

Table 7
The P2P mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
Unstructured	Each peer is randomly connected to a fixed number of peers	Is reliable in terms of query correctness, single point of failure and false-positive errors. It can tolerate node dynamicity, and scalable since complexities are low and load is distributed	Does not fully support the dynamic and multi-attribute range queries and suffers from the network-wide broadcast storm problem
Structured	Is based on distributed indexing service	Can support resource diversity and provides scalability, robustness, and self-organizing; also support dynamic attribute queries	Suffers from high traffic, low scalability and false-positive errors. It not rapidly changes the information and not fully supports the multi-attribute range queries
Super to Peer	Grid nodes are divided into clusters, each having one or more super-peers	Provides autonomy, load-balancing and fault-tolerant. Also support range and multi-attribute queries	Suffers from difficult implementation, false-positive errors, complex clustering procedure, and single point of failure in each cluster. Dynamic attribute queries not supported
Hybrid	Combines two mechanisms and benefit from the advantageous of them	Provides high reliability	Suffers from high overhead

3.4. Hierarchical mechanisms

In this section, first, the hierarchical mechanisms of resource discovery and their basic properties are described. Then, several popular structured hierarchical mechanisms of resource discovery are discussed. Finally, the discussed hierarchical mechanisms are compared and summarized.

3.4.1. Overview of hierarchical mechanisms

In hierarchical mechanisms the information of resources is updated, sorted and indexed under a set of hierarchical nodes like Globus resource discovery mechanism.[12] At the closer level of the resources, the proximity clusters of the resources are being built, while the higher layer connects these clusters, the connectivity of the distributed system is provided (Papadakis et al., 2007). In these approaches, queries are processed hierarchically since servers have been organized hierarchically so that each server is responsible for partitions of resource information. These mechanisms guarantee low search delay due to the use of multi-layer architecture. However, some servers suffer more from loads than others. They also have static partitioning scheme. Moreover, they do not work well when value of attributes changes rapidly since it takes a long time until the resource information are received by the upper nodes (Harvey et al., 2003).

3.4.2. Popular hierarchical mechanisms

In 2001 MDS-2 (Czajkowski et al., 2001) provides a configurable information provider component called Grid resource information service (GRIS) and a directory component named Grid index information service (GIIS). In MDS-2, dynamic and static resource information is provided by the GRIS service. The provided resource information is stored in aggregated directories using GIIS. An information provider registers itself to a directory regarding the local and virtual organization specific policies. The provider then updates its resource status in its registered directory. If a resource provider does not update its resource information for a predefined period, the directory assumes that the provider has become unavailable and deletes the provider from its directory. As long as a provider exists in a directory, it is included in results for relevant discovery queries. MDS-2 uses the LDAP as a uniform means of storing system information from a rich variety of system components, for constructing a uniform namespace for resource information across a system that may consist of many organizations, and for query processing (Zhang et al., 2007). MDS-2 also supports secure data access through the use of Grid Security Infrastructure (GSI) credentials. Furthermore, it supports multi-attribute queries and provides high scalability. But MDS-2 servers have significant load and memory usage problems when run for prolonged periods without restarting. Also, the time to answer a query can vary in different clients (Schopf et al., 2006).

Elmroth and Tordsson (2005) proposed a new Grid resource discovery mechanism and job submission system which is a complete Grid management middleware. In this mechanism a WSRF[13]-based metascheduler is designed, which performs task allocation to resources. Its Grid service receives single task requests, which cause a resource discovery to occur prior to execution. The authors take advantages of many existing tools by utilizing and extending them. For the resource discovery phase, they use Grid Laboratory Uniform Environment (GLUE) project which provides information related to the resources. The server side module which is called job submission module is composed of seven components. Information Finder is the most important component which discovers Grid resources, indexes resource information and provides information about them. This index server is implemented hierarchically. When a query is received, the Information Finder performs resource discovery by querying each index server (Elmroth and Tordsson, 2005). The Information Finder also retrieves usage policies, allowing it to discard resources where the user is not authorized to submit jobs. This mechanism reduces the probability of bottleneck problem. But it cannot solve single point of failure since failure of a server in the hierarchical organization may result in a large part of the resources to be excluded from the queries. On the other hand, since the resource information is stored without any hash function and since it is verified by contacting the candidates directly, the system supports all multi-attribute, range and dynamic attribute queries (Cokuslu et al., 2010). In addition it supports jobs with hard deadlines.

Ramos et al. (2006) proposed a web service for resource discovery in Grids based on Globus Toolkit. They proposed a hierarchical topology in which the Grid environment is divided into virtual organizations (VO) (Ramos et al., 2006). In each VO, there are master and slave nodes. The master nodes are responsible for updating the resource information and the slave nodes are responsible for retrieving resource information from each machine which composes the Grid system (Ramos et al., 2006). Each master machine has a resource that describes the slave machine using XML format. The XML file contains IPs and names of slaves. The resource discovery is realized by the preparation of a configuration resource which includes the requested resource information (Ramos et al., 2006). Since this mechanism is a hierarchical web service based system, it decreases the possibility of bottleneck problem. But, failure of a master machine may result in a large number of slave machines becoming invisible to the system since the resource discovery task is managed by master machines. This mechanism supports all multi-attribute, range and dynamic-attribute queries since the resource discovery is fulfilled without any hashing function (Cokuslu et al., 2010).

Yulan et al. (2007) proposed a layered hierarchical Grid resource discovery method. The first layer is composed of resource nodes. The second layer is resource information layer which consists of the nodes to store resource information. Each node in the first layer is linked to a resource information node in the second layer. The nodes in the resource information layer form virtual organizations and each virtual organization has a super node. The size of each virtual organization in the resource information layer is restricted to a predefined value in order to ensure scalability. Resource discovery process is done by four phase: submission, sorting, searching and location. The proposed mechanism is compared with Exhaustive and Lumped models. The Exhaustive model searches all the resource information nodes and the Lumped model uses a single point for resource discovery. As Yulan et al. explained, the hierarchical model not only outperforms the other two models in terms of resource discovery time but it also does not exhibit performance problems of the other two models. However, single point-based resource discovery in the Lumped Model can become a bottleneck and source of failure. Also, blind search discovery in the Exhaustive Model did not scale when the resources are increasing. This mechanism has better performance than the other two mechanisms in terms of resource discovery time and does not have performance problems of the other two models. This mechanism ensures scalability and reliability; in addition it supports multi-attribute, range and dynamic attribute queries. But it suffers from high complexity and security problems.

Huedo et al. (2009) proposed recursive architecture for hierarchical Grid resource management. The basic idea is to forward user requests to another domain when the current one is overloaded. This mechanism creates a hierarchy of Grids by abstracting remote Grids and presenting them to the local system in the same way as a local

[12] www.globus.org
[13] Web Services Resource Framework.

resource. Also the potential benefits of recursive architecture for hierarchical Grid resource management are shown. To better illustrate them, they presented some implementation details, real use cases and experiences. The main strength of the proposed mechanism is the use of Globus interfaces for Grid infrastructure access and federation. This makes existing tools ready to be used without modifications and enormously simplifies interoperation. This mechanism has better performance in terms of autonomy, scalability, deployment and security (Huedo et al., 2009). But this architecture suffers from high complexity and overheads.

In 2010, a new method has been introduced which used tree mechanism in resource discovery by transforming all resource attributes to bitmap representations by Chang and Hu (2010). This mechanism uses two bitmaps called the 'local resource bitmap' and 'index bitmap'. The local resource bitmap registers information about the local resources of nodes and the index bitmap registers the information about its children nodes which exist in the nodes that have child. These nodes utilizes AND operation to discover the resources required by the users. User requests are forwarded to Indexing Servers (IS) to find whether there is a matched resource. If it can find matched resources in a node, the request will be forwarded to the children nodes. If there are no matched resources, the search will forward up the tree until it reaches the root (Chang and Hu, 2010). Therefore, this mechanism can reduce the traffic and improve the resource discovery efficiency by using this tree architecture. The obtained results showed that the number of visited nodes and performed operations, as a result of query forwarding, are smaller than MMO algorithm and flooding based approach (Marzolla et al., 2005, 2007). Also, the cost of update in this algorithm is lower. While attributes were kept about resources and services, the mechanism showed that the attributes were stored as bitmaps which each bit in the map indicates what attribute a given resource exhibits. However, in this mechanism the extensibility was unclear and there was no test case presented where a resource with an attribute not previously known in the bitmap was published. Also, this mechanism suffers from single point failure. The testing itself was inconsistent thus making the results questionable at best. In one case, the solution was executed with 100 queries while a related solution was tested with 300 queries. Finally, while it was stated that the solution had durability against the constant addition and removal of tree nodes, a test case supporting this claim was not presented (Goscinski and Brock, 2010).

Ebadi and Khanli (2011) introduced a new distributed and hierarchical mechanism for service discovery in a Grid environment for improving fault tolerance and speed of service discovery. The architecture of resource discovery in this mechanism has five layers: client and service, institution, organization, domain, and root layers. In this architecture all nodes in one level which have the same parent, send requests to each other directly. To improve fault tolerance in this mechanism, after the requested service is found, the process of service discovery continues to find several instances of the requested service. This action is done to improve fault tolerance and speed up the service replacement time with another, when the service faces a problem or the service provider leaves the Grid. Furthermore it reduces the traffic and routes requesting when facing a fault. Also for speeding up service discovery, the nodes in the same level which have the same parent send queries to each other directly. The authors showed that the service discovery mechanism gets a good efficiency, and consistency. Also, in comparison to DST-based mechanism (Xiao-Hua et al., 2006) the proposed mechanism improves the fault tolerance of services more than 28% (Ebadi and Khanli, 2011). But the traffic overhead is more than the DST-based mechanism. However, this mechanism is inefficient when the scale of Grid rapidly increases. Also, the system suffers from single point of failure at the tree root.

3.4.3. Summary of hierarchical mechanisms

Hierarchical mechanisms reduce the network's traffic, offer some proximity search capabilities which are raised from the system's design and are more scalable than centralized systems. But, a major drawback is the inability to provide a fast response to the queries that are not answered at the lower level. While the super peer mechanisms are more effective in very large Grids, the hierarchical mechanisms are practical for small and medium sized Grids. Also it might take a long time for resource information to be updated from the leaf nodes to the root node. In addition, single point of failure in tree root decreases the system fault tolerating feature. Table 8 provides the summary of main properties of hierarchical mechanisms.

3.5. Agent-based mechanisms

In this section, the agent-based mechanisms of resource discovery and their basic features are described. Then, most popular agent-based mechanisms are discussed. Finally, the discussed agent-based mechanisms are compared and summarized in Section 3.5.3.

Table 8
The hierarchical mechanisms and their properties.

Mechanism	Main idea	Advantages	Disadvantages
MDS-2 (Czajkowski et al.)	Resource information is provided by the GRIS service and stored in aggregated directories using GIIS	Supports multi-attribute queries and provides high scalability	Has a load and memory usage problem
Elmroth and Tordsson	Provides s a metascheduler based on WSRF to perform task allocation to resources	Reduces the probability of bottleneck, supports multi-attribute, range and dynamic attribute queries	Cannot solve single point of failure problem
Ramos and Melo	Is based on hierarchical topology in which the Grid environment is divided into virtual organizations	Decreases the possibility of bottleneck problem, supports all multi-attribute, range and dynamic-attribute queries	Single point of failure exits in each VO
Yulan et al.	Follows 3-layered hierarchical architecture	Ensures scalability, reliability and supports multi-attribute, range and dynamic attribute queries	Suffers from high complexity and security problems
Chang and Hu	Is based on tree and uses two bitmaps called the 'local resource bitmap' and 'index bitmap'	Reduces traffic and cost of update.	Suffers from extensibility and single point of failure
Ebadi and Khanli	Presents five layers: client and service layer, institution layer, organization layer, domain layer, and root layer	Provides efficiency, fault tolerance and consistency	Is inefficient when the scale of Grid rapidly increases and suffers from single point failure

3.5.1. Overview of agent-based mechanisms

Agent-based technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems in open and dynamic environments (Krauter et al., 2002). Agent architectures are designed to exhibit autonomy, decentralized coordination, and complex distributed behaviors in highly dynamic environments such as Grids (Foster et al., 2004; Jennings, 2001). In agent-based mechanisms, active code fragments are sent across the machines in the Grid and those fragments are locally interpreted at each machine. Furthermore, the agent makes resource discovery decisions based on its logic (Tan, 2009).

3.5.2. Popular agent-based mechanisms

In this section several agent-based mechanisms of resource discovery in Grid environments are provided and discussed as follow:

Kakarontzas and Savvas (2006) described a resource discovery and selection system for dynamic Grids based on agent. In this mechanism the client agents act on behalf of Grid users, and search for resources in the network. This mechanism consists of two phases: resource discovery phase and resource selection phase. In the resource discovery phase, resources that do not satisfy static resource requirements are filtered out. Then, in the resource selection phase, requesters negotiate directly with providers to determine the current state of the remaining resources and select those suitable for the job's execution. After potentially suitable resources are discovered, client agents carry out negotiations directly with agents representing local resource management systems (Kakarontzas and Savvas, 2006). This mechanism reduces load index requirements. Also use of a large number of resource management agents in this algorithm removes bottleneck problem and makes the algorithm more scalable. Moreover, since queries are flooded with TTL parameter, it may result in false-positive errors. However, in most of the agent-based mechanisms multi-attribute and range queries are supported due to the lack of hashing. Moreover, since the up-to-date nodes' statuses are obtained by communicating with resource nodes, dynamic-attribute queries are also supported.

Mobile agent is a new network technology which is finding its way into many commercial applications areas. Mobile agent is an intelligent agent with mobility that could be moved independently from one host to another on the network, and complete specific tasks on behalf of the user, such as searching, filtering and collecting information, even do commercial activities on behalf of users. Mobile agent has the autonomy, responsiveness, initiation, communication and mobility and other characteristics (Lei et al., 2010). Zhao et al. (2007) proposed mobile agent-based resource management in Grid systems. In this mechanism a requester creates a mobile agent and dispatches it to a desired remote machine on the network. The search criteria are defined in the agent that carries the related code to the remote machine. Then, the results are dispatched to the sender. Mobile agent-based resources information collection method transmits a small quantity of running code and status to resource nodes, then analyzes and predicts the resources function, and brings the results back to the information server. Since transmitting a large quantity of resources' dynamic information back to the resources information server is avoided, the network load is decreased. This mechanism reduces the load on network, put the code design at risk, and provides a robust application over unreliable networks. Furthermore, it reduces the cost of the system and improves the efficiency of resource management. But each machine needs an agent server and protection against the malicious code and general code so difficult results in the security are decreased. These disadvantages will affect the system response time (Zhao et al., 2007).

Toninelli et al. (2008) proposed a semantic-based discovery mechanism to support mobile context-aware service discovery in 2008. The mechanism uses the context-awareness on basis of user/device/service profile metadata which is called AIDAS (Adaptable Intelligent Discovery of context-Aware Services). AIDAS is a middleware that exploits semantic techniques to perform context-aware discovery of services. It uses semantics-based metadata to describe the properties and characteristics of the services and clients involved in discovery. The design of the AIDAS middleware tackles the challenge of making semantic-based discovery viable even to resource-constrained devices. To fit this purpose, AIDAS integrates several middleware facilities capable of adapting semantic support to the different characteristics of mobile devices and of providing mobile devices with visibility on semantic functionalities hosted by nearby devices. This mechanism enables knowledge sharing in open dynamic systems and allows an efficient reasoning on context information with well-defined declarative semantics. Also, AIDAS is able to find all services whose capabilities have a semantic relation with requested capabilities, according to the service ontology (Toninelli et al., 2008). When compared to a centralized system, this mechanism does not suffer from the single point of failure. But, it is limited to the Java language and uses the large volume of storage resources. Also, the system suffers from high complexity. In addition, the matching response time must be improved.

A novel semantic-supported and agent-based decentralized Grid resource discovery mechanism is proposed by Han and Berry (2008). This mechanism used a heuristic algorithm to find neighboring resource agents without overhead of negotiation, and allows individual resource agents to semantically interact with neighbor agents based on local knowledge rather than global information. An agent only needs to know who he is connecting to and what kind of resource is carried by neighbor agents. Through semantic similarity calculation, individual resource agents have flexible matching functions and thus dynamically discover resources required by tasks in an efficient way. The algorithm ensures the resource agent's ability to cooperate and coordinate with neighbor knowledge requisition for flexible problem solving. The algorithm is evaluated by investigating the relationship between the success probability of resource discovery and semantic similarity under different factors. The experimental results showed that the algorithm could flexibly and dynamically discovers resources and therefore provides a valuable addition to the field. In addition, increasing the level of the task complexity resulting in decreasing the success probability, the higher task load caused the higher success probability and the larger the network scale caused the higher success probability (Han and Berry, 2008). But in this mechanism discovering the resources is slow and more query traffic occurs.

Multi-agent systems (MAS) are relatively new software paradigms that are widely accepted in several application domains to address large and complex tasks (Cavalcante et al., 2012). MAS are a loosely coupled network of software agents that cooperate to solve problems that may be beyond the individual capacities or knowledge of each particular agent (Tan, 2009). In MAS, computational resources are distributed across a network of interconnected agents. MAS efficiently retrieve, filters, and globally coordinate information from sources that are spatially distributed. Agents in MAS need to use resources provided by other agents. Tan et al. (2010) proposed a multi-agent mechanism to discover resources in Grid environments. The aim of this mechanism is to disseminate Grid resources and spatially map them on the Grid according to their semantic classification, in order to gather a consistent number of resources of the same class in a restricted region of

the Grid. It is assumed that the resources have been previously classified into a number of communities of Interests, according to their semantics and functionalities. This scheme uses the random movements and operations of a number of mobile agents that travel the Grid using the peer to peer interconnections. This mechanism is inspired by agent-based systems where swarm intelligence emerges from the collective behavior of very simple mobile agents, and a complex overall objective is obtained. In this mechanism, each mobile agent can pick a number of resources on a Grid host, carry such resources while moving from host to host, and deposit them on another Grid host (Cavalcante et al., 2012; Tan, 2009). When compared to a centralized system, this mechanism does not suffer from the single point of failure problem. Furthermore, it has less performance bottlenecks or resource limitations. In addition, discovering the resources needs short time. But the mechanism has high complexity because it must support MAS. In 2012 another multi-agent-based brokering protocol for Grid resource discovery was proposed by Kang and Sim (2012). In this mechanism three types of agents (user agent, provider agent, and broker agent) are used. Each broker agent connects user agents to provider agents using the connection algorithm which mainly consists of 4 stages: selection, evaluation, filtering, and recommendation. In the recommendation stage, two kinds of recommendation approaches (circular approach and multicast approach) are used for making recommendations to the user agents that failed to be matched to provider agents. The connection procedure is done by four stages: Selection, Evaluation, Filtering, and Recommendation. Authors showed that their mechanism has good performance in terms of scalability and adaptability. But the response time could be high for a large network in high workload and the complexity of the mechanism is considerable.

3.5.3. *Summary of agent-based mechanisms*

Agent based systems are attractive in Grid systems because of their autonomous property. They have capabilities to determine new migration sites according to their migration policies for the distribution of resource discovery queries. In fact, an agent is flexible, mobile, and autonomous. Integrating mobile agent technology and Grid technology to make most of the advantages of the both will considerably reduce the cost of communications in Grid resources discovery. In these approaches, since the underlying network topology is unstructured, the system does not suffer from bottleneck. Agent based communication paradigms have shown enormous potential for operating in unpredictable, metamorphic environments, such as mobile computing networks. Moreover, the distribution of queries is not limited by a TTL value, which removes false-positive errors. Mobile agent technology also has a few disadvantages. One of its disadvantages is that it requires an agent server on each participating machine. The other disadvantages are protection against the malicious code and general code and data security. These disadvantages will affect the system response time. However, most of agent-based mechanisms easily support range, multi-attribute and dynamic-attribute queries as they respond to queries without any discrete mapping function such as hashing. Table 9 summarizes the agent-based mechanisms and their properties.

4. Results and comparison

Here, a comparison of mentioned mechanisms of resource discovery is discussed. In this paper resource discovery mechanisms have been divided into five main categories; centralized, decentralized, peer to peer, hierarchical, and agent-based.

In centralized mechanisms the authorization and security issues are handled easily as they are administrated from a central system. But they are not scalable and have a single point of failure. The central database suffers from a high computational overhead and overall performance reduction. Furthermore, these mechanisms tolerate node dynamicity. Also these mechanism support range and multi-attribute queries since queries are resolved in centralized servers without any hashing but because of periodic updates, the dynamic attribute queries are not completely supported. In addition, parallel access to the central system is very limited and the response time could be very high for a high workload.

The main drawback of centralized mechanism is its central database. In order to overcome this drawback, the global database must be replicated. This idea is basis of decentralized mechanisms. Decentralized mechanisms can remedy the problem of centralized mechanisms. These mechanisms absolutely balance loads, scale well, and can tolerate the failure of distributed operations. Furthermore, these mechanisms tolerate node dynamicity. Also, these mechanisms support range queries and multi-attribute queries since queries are resolved within hierarchically distributed servers without any hashing. But additional overhead is created by managing the network architecture. Also some of the unstructured decentralized mechanisms (such as Iamnitchi and Foster, 2001) use undesirable methods to discover the resource like flooding and broadcasting.

Peer to Peer mechanisms have been divided into four main categories: unstructured P2P mechanisms, structured P2P mechanisms, hybrid P2P mechanisms and super to peer mechanisms. Unstructured peer-to-peer mechanisms are very robust and reliable

Table 9
The agent-based mechanisms and their properties

Mechanism	Main idea	Advantages	Disadvantages
Kakarontzas and Savvas	Client agents act on behalf of Grid users, and search for resources in the network by use of two phases	Multi-attribute, range and dynamic attribute queries are supported. It provides high scalability and there is not any bottleneck	False-positive errors may occur
Zhao et al.	A requester creates a mobile agent and dispatches it	Offers low network load and cost	Suffers from low security and high response time
Toninelli et al.	Uses semantic-based metadata to describe the properties and characteristics of the services	Does not suffer from the single point of failure problem	Uses the large volume of storage resources, high complexity
Han and Berry	Uses a heuristic algorithm to find neighboring resource agents	Offers high flexibly and dynamicity, low negotiation overhead	Suffers from low discovering speed and high traffic
Tan et al.	Disseminates Grid resources and spatially map them on the Grid according to their semantic classification	Does not suffers from the single point of failure problem, has less performance bottlenecks	Suffers from high complexity
Kang and Sim	Three types of agents (user agent, provider agent, and broker agent) to discover resources are used	Provides scalability and adaptability	Suffers from low response time and high complexity

Table 10
Factors of comparison between resource discovery mechanisms.

	Centralized	Decentralized	Peer to Peer	Hierarchical	Agent Based
Scalability	Low	High	Structured: high, others: low	Medium	Medium
Security	High	Low	Low	Medium	Medium (mobile agents: low)
Dynamicity	Tolerate node dynamicity	Tolerate node dynamicity	Except unstructured, others do not tolerate dynamicity	Tolerate node dynamicity	Tolerate node dynamicity
Reliability	Reliable in terms of query correctness, but not reliable in terms of single point of failure	Reliable in terms of query correctness, more reliable in terms of single point of failure	Reliable in terms of query correctness, more reliable in terms of single point of failure	Reliable in terms of query correctness, less reliable in terms of single point of failure	Reliable in terms of query correctness, more reliable in terms of single point of failure
Range queries	Supported but network traffic to certain nodes are increased dramatically	Supported	Most of them supported	Supported	Supported
Multi-attribute queries	Supported	Supported	Most of them supported	Supported	Supported
Dynamic-attribute queries	Not supported because of the periodic updates	Not supported because of the periodic updates	Supported by all mechanisms except super peer	Not supported because of the periodic updates	Most of them supported
Overhead and complexity	Low	Medium	Medium	High	High
Response time	High	Medium	High	High	Medium
Main advantageous	Simplicity	Scalability	Fault tolerating	Scalability	Fault tolerating and robustness
Main disadvantageous	Single point of failure	Security	Security and response time	Complexity and response time	Complexity

in terms of query correctness and single point of failure. In addition they can tolerate node dynamicity and can easily support range, multi-attribute and dynamic-attribute queries. Moreover, these approaches can easily handle dynamicity of the Grid since both resources and indexing nodes are distributed to the network. But they cannot guarantee the number of hops taken in order to deliver service to requester. Also they suffer from false-positive errors caused by the usage of TTL limitations. Furthermore, these mechanisms cannot guarantee that results will be returned if they exist in the network because of time-to-live field, also they suffer from the network-wide broadcast storm. But these mechanisms are popular in the Internet community in spite of their disadvantages. Structured P2P resource discovery mechanisms are more scalable than unstructured ones, in terms of traffic load, but need strong self-organizing mechanisms in order to maintain their fixed structure. Structured resource discovery mechanisms are prone to node failure, and unpredictable node departures (Wang et al., 2009). Also, membership management hinders their usability in highly dynamic systems, and these mechanisms are difficult to be implemented and use the rich resource description. Furthermore, the level of robustness in these mechanisms is very low and needs periodic updates. Due to periodic updates, they do not fully support dynamic attribute queries. On the other hand, in most methods since the queries are distributed to the network by following a defined path in the topological structure, failure of a node may result in loss of queries in the network. This causes the single point of failure. As the query is relayed to the end of its search domain, the use of structured query routing mechanisms can remove false-positive errors. A third category of P2P mechanisms is super to peer mechanism. The super-peer mechanism can be adopted in large-scale Grids because of its efficient implementation of the information service. In a super-peer mechanisms the communications take place only among super-peers, therefore it is the most appropriate mechanism to ensure extensibility and scalability of the system (Mastroianni et al., 2005). Moreover, since the queries are resolved by checking the index tables, this approach can easily support range queries and multi-attribute queries. But because the resource information is collected by the super-peers at periodic intervals, this method does not support dynamic-attribute queries in its nature. But these mechanisms suffer from the difficult implementation and use of the rich resource description. Furthermore, due to the disjointed information source and the point of storage, the network load in dynamic environments is considerable. Also, the clustering procedure may become more complicated. In addition, the systems suffer from the bottlenecks when the number of request for the super-peer is very large. Finally, single point of failure exists in each cluster. A hybrid mechanism can be defined as a mixture of two mechanisms which use their advantages. These mechanisms almost have high overhead but are reliable. Briefly, P2P mechanisms provide high scalability, can tolerate node dynamicity and are reliable in terms of query correctness and single point of failure; but suffer from security issues and response time.

Hierarchical mechanisms do not scale well in Grid systems in which frequent updates and large numbers of requests exist. In addition, single point of failure in tree root decreases the system fault tolerating feature. But these approaches offer better capability and failure tolerance than centralized mechanisms due to the fact that resources are partitioned on different hierarchical servers. Furthermore, these mechanisms can reduce the network's traffic because of system's design and provide some proximity to search capabilities. These mechanisms are reliable in terms of query correctness but less reliable in terms of single point of failure. Also these mechanisms support range and multi-attribute queries but because of periodic updates, dynamic attribute queries are not supported completely.

Agent based mechanisms require an agent server on each participating machine. In these mechanisms the response time of query request is almost high. Also the bottleneck problem is removed since the load on nodes is distributed. Furthermore these mechanisms are protected against the malicious and general code and data security. Furthermore mobile agent-based mechanisms reduce the cost of communications. However, these mechanisms are reliable in terms of query correctness and single point of failure; but they suffer from high complexity and single point of problems. Also most of these mechanisms easily support range and multi-attribute queries since they are processed within the nodes without any discrete mapping function such as hashing. The dynamic-attribute queries are also supported by the use of agent technology, which allows the resources to send updates about their dynamic-attributes continuously instead of at periodical

intervals. Table 10 summarizes the comparison among resource discovery mechanisms in Grid systems and provides taxonomy among resource discovery mechanisms in terms of scalability, security, dynamicity, reliability, query supporting, complexity and response time.

5. Open issues

It has been observed that there is no single mechanism that addresses all issues involved in resource discovery. For example some strategies consider providing reliability, scalability, dynamicity and load balancing while some totally ignore these issues. Furthermore, most of the techniques included in this survey have been used with simulation to evaluate and test the proposed mechanisms. As a next step these mechanism can be tested in real world scenarios to provide a very realistic results.

Additionally, predicting the performance of the resources that are geographically distributed in both location and information will become a challenging problem. Also in a fully decentralized resource discovery architecture, which has no central server, performing the resource discovery efficiently and making the best resource scheduling is very challenging. Furthermore, resource management with a fully decentralized resource discovery mechanism is much more challenging. In addition, another interesting point of future study is to investigate the effects the size would have on systems performance in a large scale environment by real Grid systems or by using some simulator such as Gridsim.

It has been observed that in the most of the mechanisms some issues such as trust, reputation and cost of service were not mentioned. Therefore a new mechanism which enables resource discovery based on trust, reputation and cost of service is very interesting. These issues are discussed and analyzed independently and sufficiently; also they are ready to be adopted to increase current discovery mechanisms' popularity and acceptance. Another interesting point is that most of the solutions proposed in the literatures do not assume specific resources where as the Grid systems can be the aggregation of specific resources such as human resources. Therefore, a mechanism to discover such resources is still very challenging.

Other interesting lines for future research and works are the development of a resource discovery mechanism for authorization and identity mapping by adopting some existing method such as Groeper et al. (2009) and Marín Pérez et al. (2011); consideration of accounting and billing concepts for effective implementation of discovery mechanism by combining existing resource discovery mechanisms with existing accounting and billing method in Grid such as Piro et al. (2009); and proposition of the multi-agent systems with learning agents to respond to the similar queries with high speed. However, in some methods with lack of security features the resource discovery mechanism can be improved with security features. Also some of the resource discovery mechanisms can be enhanced to discover resources under cloud computing environment. Moreover, in some cases the resource discovery processes can be improved to search the resources with low cost. Failure management and task migration features are important concepts which get low attention in current discovery mechanisms; therefore they can be added to existing method for improving their efficiency. Some methods are only studied in data or computational Grid; they can be adapted to each other to improve their application domains.

Mobile Grid combines Grid and mobile computing and supports mobile users and resources in a seamless, transparent, secure and efficient way (Li and Li, 2011). When users of mobile Grid need to access any services or resources, they may face some issues such as congestion due to the limited bandwidths, network disconnection, and the signal attenuation caused by users' mobility. Because of users and resources mobility, mobile resources discovery is other main concern which can be considered in future researches.

6. Conclusion

In this paper, we have surveyed the past and the state of the art mechanisms in Grid resource discovery. Furthermore, we introduced taxonomy of Grid resource discovery mechanisms so far. We have divided the Grid resource discovery mechanisms into five main categories: Centralized, Decentralized, Peer-to-Peer, Hierarchical, and Agent-Based. For each of these classes, we reviewed and compared several proposed mechanisms. The centralized mechanisms do not scale well, do not support dynamic-attribute queries, and are not reliable in terms of single point of failure. Decentralized mechanisms eliminate single point of failure problem of centralized mechanisms and absolutely balance loads, and can tolerate the failure of distributed operations. Agent-based mechanisms have some advantages but they have high complexity. Hierarchical mechanisms do not scale well and have a single point of failure. Structured P2P mechanisms support multi-attribute queries but suffer from the network-wide broadcast storm. However, unstructured P2P mechanisms suffer from the network-wide broadcast storm and false-positive errors. Therefore a specific mechanism to provide all mentioned issues will become a challenging problem and are interesting lines for future research and work.

References

Akbari Torkestani Javad. A distributed resource discovery algorithm for P2P grids. Journal of Network and Computer Applications 2012;35(6):2028–36.

Ali HA, Ahmed MA. HPRDG: A scalable framework hypercube-P2P-based for resource discovery in computational Grid. In: Paper presented at the 2012 22nd international conference on computer theory and applications (ICCTA); 13–15 October 2012.

Andrzejak Artur, Xu Zhichen. Scalable, efficient range queries for grid information services. In: Paper presented at the proceedings of the second international conference on peer-to-peer computing; 2002.

Ashri Ronald. Agent middleware technologies: a review of Jini and JXTA; 2003.

Balasangameshwara Jasma, Raju Nedunchezhian. A hybrid policy for fault tolerant load balancing in grid computing environments. Journal of Network and Computer Applications 2012;35(1):412–22.

Bellwood Tom. UDDI version 2.04 API specification: UDDI committee specification; 2002.

Belqasmi Fatna, Glitho Roch, Dssouli Rachida. An overlay network for autonomous information discovery in the post-composition registries of ambient networks. Journal of Network and Computer Applications 2011;34(2):697–707.

Benson, Edward, Wasson, Glenn, & Humphrey, Marty. (2006). Evaluation of UDDI as a provider of resource discovery services for OGSA-based grids. In: Paper presented at the proceedings of the 20th international conference on Parallel and distributed processing, Rhodes Island, Greece.

Beverly Yang B, Garcia-Molina H. Designing a super-peer network. In: Paper presented at proceedings 19th international conference on the data engineering, 2003; 5–8 March 2003.

Binzenhöfer Andreas, Schnabel Holger. Improving the performance and robustness of Kademlia-based overlay networks. In: Braun T, Carle G, Stiller B, editors. Kommunikation in Verteilten Systemen (KiVS). Berlin, Heidelberg: Springer; 2007. p. 15–26.

Brocco Amos, Malatras Apostolos, Hirsbrunner Béat. Enabling efficient information discovery in a self-structured grid. Future Generation Computer Systems 2010;26(6):838–46.

Cai Min, Frank Martin, Chen Jinbo, Szekely Pedro. MAAN: a multi-attribute addressable network for grid information services. Journal of Grid Computing 2004;2(1):3–14.

Castellà, Damià, Giné, Francesc, Solsona, Francesc, Lluis Lérida, Josep. Analyzing locality over a P2P computing architecture. Journal of Network and Computer Applications, in press. http://dx.doi.org/10.1016/j.jnca.2012.12.032.

Cavalcante Rodolfo Carneiro, Bittencourt Ig Ibert, da Silva Alan Pedro, Silva Marlos, Costa Evandro, Santos Robério. A survey of security in multi-agent systems. Expert Systems with Applications 2012;39(5):4835–46.

Chang Ruay-Shiung, Hu Min-Shuo. A resource discovery tree using bitmap for grids. Future Generation Computer Systems 2010;26(1):29–37.

Chen Shiping, Zhang Zhan, Chen Shigang, Shi Baile. Efficient file search in non-DHT P2P networks. Computer Communications 2008;31(2):304–17.
Christensen Erik, Curbera Francisco, Meredith Greg, Weerawarana, Sanjiva. Web Services Description Language (WSDL) 1.1; 2001.
Cokuslu Deniz, Hameurlain Abdelkader, Erciyes Kayhan. Grid resource discovery based on centralized and hierarchical architectures. International Journal for Infonomics 2010;3(1):227–33.
Cox, Russ, Muthitacharoen, Athicha, & Morris, Robert. Serving DNS using a peer-to-peer lookup service. In: Paper presented at the revised papers from the first international workshop on peer-to-peer systems; 2002.
Czajkowski K., Fitzgerald S., Foster I., Kesselman C. Grid information services for distributed resource sharing. In: Paper presented at proceedings 10th IEEE international symposium on the high performance distributed computing; 2001.
Dabek F, Brunskill E, Kaashoek MF, Karger D, Morris R, Stoica I., et al. Building peer-to-peer systems with chord, a distributed lookup service. In: Paper presented at proceedings of the eighth workshop on the hot topics in operating systems, 2001; 20–22 May 2001.
Dabek Frank, Kaashoek M, Frans Karger David, Morris Robert, Stoica Ion. Wide-area cooperative storage with CFS. In: Paper presented at the proceedings of the eighteenth ACM symposium on operating systems principles, Banff, Alberta, Canada; 2001.
Deng Yuhui, Wang Frank, Ciura Adrian. Ant colony optimization inspired resource discovery in P2P Grid systems. The Journal of Supercomputing 2009;49(1):4–21, http://dx.doi.org/10.1007/s11227-008-0214-0.
Di Stefano Antonella, Morana Giovanni, Zito Daniele. A P2P strategy for QoS discovery and SLA negotiation in grid environment. Future Generation Computer Systems 2009;25(8):862–75.
Doguc Ozge, Emmanuel Ramirez-Marquez Jose. An automated method for estimating reliability of grid systems using Bayesian networks. Reliability Engineering & System Safety 2012;104(0):96–105.
Ebadi Saeed, Khanli Leyli Mohammad. A new distributed and hierarchical mechanism for service discovery in a grid environment. Future Generation Computer Systems 2011;27(6):836–42.
Elmroth E, Tordsson J. An interoperable, standards-based grid resource broker and job submission service. In: Paper presented at the 2005. First international conference on e-science and grid computing; 1–1 July 2005.
Erdil DCenk. Simulating peer-to-peer cloud resource scheduling. Peer-to-Peer Networking and Applications 2012;5(3):219–30, http://dx.doi.org/10.1007/s12083-011-0112-8.
Fitzgerald S, Foster I, Kesselman C, von Laszewski G, Smith W, Tuecke S. A directory service for configuring high-performance distributed computations. In: Paper presented at the the sixth IEEE international symposium on high performance distributed computing, 1997. Proceedings; 5–8 August 1997.
Flocchini Paola, Nayak Amiya, Xie Ming. Hybrid-chord: a peer-to-peer system based on chord. In: Ghosh RK, Mohanty H, editors. Distributed computing and internet technology, vol. 3347. Berlin, Heidelberg: Springer; 2005. p. 194–203.
Foster I, Jennings NR., Kesselman, C. Brain meets brawn:why grid and agents need each other. In: Paper presented at the proceedings of the third international joint conference on autonomous agents and multiagent systems, 2004. AAMAS 2004; 23 July 2004.
Foster Ian, Kesselman Carl. The grid: blueprint for a new computing infrastructure. Morgan Kaufmann; 1999.
Foster Ian, Kesselman Carl, Tuecke Steven. The anatomy of the grid: enabling scalable virtual organizations. International Journal of High Performance Computing Applications 2001;15(3):200–22, http://dx.doi.org/10.1177/109434200101500302.
Fouad Butt, Syed Saadat Bokhari, Abdolreza Abhari, Alexander Ferworn. Scalable grid resource discovery through distributed search. International Journal of Distributed and Parallel Systems 2011;2(5):1–19.
Fraigniaud Pierre, Gauron Philippe. D2B: a de Bruijn based content-addressable network. Theoretical Computer Science 2006;355(1):65–79, http://dx.doi.org/10.1016/j.tcs.2005.12.006.
Goscinski Andrzej, Brock Michael. Toward dynamic and attribute based publication, discovery and selection for cloud computing. Future Generation Computer Systems 2010;26(7):947–70.
Groeper Ralf, Grimm Christian, Makedanz Siegfried, Pfeiffenberger Hans, Ziegler Wolfgang, Gietz Peter, et al. A concept for attribute-based authorization on D-Grid resources. Future Generation Computer Systems 2009;25(3):275–80.
Haas Zygmunt J, Pearlman Marc R. The performance of query control schemes for the zone routing protocol. IEEE/ACM Transactions on Networking 2001;9(4):427–38, http://dx.doi.org/10.1109/90.944341.
Hameurlain Abdelkader. Evolution of query optimization methods: from centralized database systems to data grid systems. In: Bhowmick S, Küng J, Wagner R, editors. Database and expert systems applications, vol. 5690. Berlin, Heidelberg: Springer; 2009. p. 460–70.
Hameurlain Abdelkader, Cokuslu Deniz, Erciyes Kayhan. Resource discovery in grid systems: a survey. International Journal of Metadata, Semantics and Ontologies 2010;5(3):251–63, http://dx.doi.org/10.1504/ijmso.2010.034048.
Han Liangxiu, Berry Dave. Semantic-supported and agent-based decentralized grid resource discovery. Future Generation Computer Systems 2008;24(8):806–12.
Hao Yongsheng, Liu Guanfeng, Wen Na. An enhanced load balancing mechanism based on deadline control on GridSim. Future Generation Computer Systems 2012;28(4):657–65.
Harvey Nicholas JA, Jones Michael B, Saroiu Stefan, Theimer Marvin, Wolman Alec. SkipNet: a scalable overlay network with practical locality properties. In: Paper presented at the proceedings of the 4th conference on usenix symposium on internet technologies and systems, vol. 4. Seattle, WA; 2003.
Hawa Mohammed, As-Sayid-Ahmad Loqman, Khalaf Loay D. On enhancing reputation management using peer-to-peer interaction history. Peer-to-Peer Networking and Applications 2013;6(1):101–13, http://dx.doi.org/10.1007/s12083-012-0142-x.
Hof Hans-Joachim, Baumgart Ingmar, Zitterbart Martina. Key exchange for service discovery in secure content addressable sensor networks. In: Braun T, Carle G, Stiller B, editors. Kommunikation in verteilten systemen (KiVS). Berlin, Heidelberg: Springer; 2007. p. 139–50.
Horng Shih-Cheng. Ordinal optimization based approach to the optimal resource allocation of grid computing system. Mathematical and Computer Modelling 2011;54(1–2):519–30.
Huedo Eduardo, Montero Rubén S, Llorente Ignacio M. A recursive architecture for hierarchical grid resource management. Future Generation Computer Systems 2009;25(4):401–5.
Iamnitchi A, Foster I, Nurmi DC. A peer-to-peer approach to resource location in grid environments. In: Paper presented at the 11th IEEE international symposium on high performance distributed computing, 2002. HPDC-11 2002. Proceedings; 2002.
Iamnitchi Adriana, Foster Ian T. On fully decentralized resource discovery in grid environments. In: Paper presented at the proceedings of the second international workshop on grid computing; 2001.
Jagadish HV, Ooi Beng Chin, Tan Kian-Lee, Vu Quang Hieu, Zhang Rong, Speeding up search in peer-to-peer networks with a multi-way tree structure. In: Paper presented at the proceedings of the 2006 ACM SIGMOD international conference on Management of data, Chicago, IL, USA; 2006.
Jennings Nicholas R. An agent-based approach for building complex software systems. Communications on ACM 2001;44(4):35–41, http://dx.doi.org/10.1145/367211.367250.
Johnston WE, Gannon D, Nitzberg B. Grids as production computing environments: the engineering aspects of NASA's Information Power Grid. In: Paper presented at the eighth international symposium on the high performance distributed computing, 1999. Proceedings; 1999.
Kakarontzas G., Savvas IK. Agent-based resource discovery and selection for dynamic grids. In: Paper presented at the 15th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises, 2006. WETICE '06; 26–28 June 2006.
Kang Jaeyong, Sim KwangMong. A multiagent brokering protocol for supporting Grid resource discovery. Applied Intelligence 2012;37(4):527–42, http://dx.doi.org/10.1007/s10489-012-0347-y.
Kaur Damandeep, Sengupta Jyotsna. Resource discovery in web-services based grids. In: Proceedings of world academy of science, engineering and technology, vol. 25; 2007. p. 284–288.
Kocak Taskin, Lacks Daniel. Design and analysis of a distributed grid resource discovery protocol. Cluster Computing 2012;15(1):37–52, http://dx.doi.org/10.1007/s10586-010-0147-2.
Kovvur RMR, Kadappa V, Ramachandram S, Govardhan A. Adaptive resource discovery models and resource selection in grids. In: Paper presented at the 2010 1st international conference on parallel distributed and grid computing (PDGC); 28–30 October 2010.
Krauter Klaus, Buyya Rajkumar, Maheswaran, Muthucumaru. A taxonomy and survey of grid resource management systems, vol. 32; 2002. p. 135–164.
Lei Zhang, Xueyong Li, Bei Ru. The research of mobile agent-based mobile grid resource discovery. In: Paper presented at the 2010 international conference on intelligent computing and cognitive informatics (ICICCI); 22–23 June 2010.
Li Chunlin, Li LaYuan. A multi-agent-based model for service-oriented interaction in a mobile grid computing environment. Pervasive and Mobile Computing 2011;7(2):270–84.
Li Hong, Liu Lu. A decentralized resource discovery based on keywords combinations and node clusters in knowledge grid. In: Paper presented at the proceedings of the intelligent computing 3rd international conference on advanced intelligent computing theories and applications, Qingdao, China; 2007.
Ludwig SA. Comparison of centralized and decentralized service discovery in a grid environment. In: Paper presented at the proceedings of 15th international conference, parallel and distributed computing and systems (PDCS), CA, USA; 2003.
Marín Pérez Juan M, Bernabé Jorge Bernal, Alcaraz Calero Jose M, Garcia Clemente Felix J, Pérez Gregorio Martínez, Gómez Skarmeta Antonio F. Semantic-based authorization architecture for Grid. Future Generation Computer Systems 2011;27(1):40–55.
Marzolla M, Mordacchini M, Orlando S. Resource discovery in a dynamic grid environment. In: Paper presented at the Sixteenth international workshop on database and expert systems applications, 2005. Proceedings; 26 August 2005.
Marzolla Moreno, Mordacchini Matteo, Orlando Salvatore. Peer-to-peer systems for discovering resources in a dynamic grid. Parallel Computing 2007;33(4–5):339–58, http://dx.doi.org/10.1016/j.parco.2007.02.006.
Mastroianni Carlo, Talia Domenico, Verta Oreste. A super-peer model for resource discovery services in large-scale grids. Future Generation Computer Systems 2005;21(8):1235–48, http://dx.doi.org/10.1016/j.future.2005.06.001.
Mastroianni Carlo, Talia Domenico, Verta Oreste. Designing an information system for Grids: comparing hierarchical, decentralized P2P and super-peer models. Parallel Computing 2008;34(10):593–611.
Maymounkov Petar, Mazières David. Kademlia: a peer-to-peer information system based on the XOR metric. In: Paper presented at the revised papers from the first international workshop on peer-to-peer systems; 2002.
Meshkova Elena, Riihijarvi Janne, Petrova Marina, Mähönen Petri. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Computer Networks 2008;52(11):2097–128, http://dx.doi.org/10.1016/j.comnet.2008.03.006.

Moreno-Vozmediano Rafael. A hybrid mechanism for resource/service discovery in ad-hoc grids. Future Generation Computer Systems 2009;25(7):717–27.
Moro Mirella M. Range query. In: Liu L, ÖZsu MT, editors. Encyclopedia of database systems. US: Springer; 2009. p. 2324–5.
Naseer A, Stergioulas LK. Web-services-based resource discovery model and service deployment on healthgrids. IEEE Transactions on Information Technology in Biomedicine 2010;14(3):838–45, http://dx.doi.org/10.1109/TITB.2010.2040482.
Papadakis Harris, Trunfio Paolo, Talia Domenico, Fragopoulou Paraskevi. Design and implementation of a hybrid P2P-based grid resource discovery system; 2007.
Piro Rosario M, Guarise Andrea, Patania Giuseppe, Werbrouck Albert. Using historical accounting information to predict the resource usage of grid jobs. Future Generation Computer Systems 2009;25(5):499–510.
Puppin Diego, Moncelli Stefano, Baraglia Ranieri, Tonellotto Nicola, Silvestri, Fabrizio. A grid information service based on peer-to-peer. In: Paper presented at the proceedings of the 11th international Euro-Par conference on parallel processing, Lisbon, Portugal; 2005.
Qiao Yi, Bustamante Fabian E. Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use. In: Paper presented at the proceedings of the annual conference on USENIX '06 annual technical conference, Boston, MA; 2006.
Ramos Tania Gomes, Melo Alba Cristina, Magalhaes Alves de. An extensible resource discovery mechanism for grid computing environments. In: Paper presented at the proceedings of the sixth IEEE international symposium on cluster computing and the grid; 2006.
Ratnasamy Sylvia, Francis Paul, Handley Mark, Karp Richard, Shenker Scott. A scalable content-addressable network. In: Paper presented at the proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, San Diego, California, USA; 2001.
Rowstron Antony, Druschel Peter. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui R, editor. Middleware 2001, vol. 2218. Berlin, Heidelberg: Springer; 2001. p. 329–50.
Ruckmani V, Sadasivam GSudha. Integrating an efficient authorization protocol with trigon-based authentication mechanism for improving grid security. In: Das V, Vijayakumar R, Debnath N, Stephen J, Meghanathan N, Sankaranarayanan S, editors. Information processing and management, vol. 70. Berlin, Heidelberg: Springer; 2010. p. 125–9.
Sanderson Paul. Identifying an existing file via KaZaA artefacts. Digital Investigation 2006;3(3):174–80.
Sarhadi Ali, Ali Yousefi, Broumandnia Ali. New method for grid computing resource discovery with dynamic structure of peer-to-peer model based on learning automata. World Applied Sciences Journal 2012;19(1), http://dxdoi.org/10.5829/idosi.wasj.2012.19.01.1597.
Schmidt Cristina, Parashar Manish. Flexible information discovery in decentralized distributed systems. In: Paper presented at the proceedings of the 12th IEEE international symposium on high performance distributed computing; 2003.
Schopf, J.M., Raicu, I., Pearlman, Laura, Miller, N., Kesselman, C., Foster, Ian T., et al. Monitoring and discovery in a web services framework: functionality and performance of globus toolkit MDS4; 2006.
Shah ShairBaz. Using P2P approach for resource discovery in Grid Computing (Master Thesis), Blekinge Institute of Technology, Sweden; 2007. MCS-2007:21.
Shen Haiying. A P2P-based intelligent resource discovery mechanism in Internet-based distributed systems. Journal of Parallel and Distributed Computing 2009;69(2):197–209, http://dx.doi.org/10.1016/j.jpdc.2008.06.010.
Singh Munindar P. Peering at peer-to-peer computing. IEEE Internet Computing 2001;5(1):4–5.
Sistla AP, Wolfson O, Chamberlain S, Dao S. Modeling and querying moving objects. In: Paper presented at the 13th international conference on data engineering, 1997. Proceedings; 7–11 April 1997.
Siva Sathya S, Syam Babu K. Survey of fault tolerant techniques for grid. Computer Science Review 2010;4(2):101–20.
SOAP Version 1.2 Part 0: Primer. In: Mitra N, Lafon Y. Editors; 2007.
Stoica Ion, Morris Robert, Karger David, Kaashoek M. Frans, Balakrishnan Hari. Chord: a scalable peer-to-peer lookup service for internet applications. In: Paper presented at the proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, San Diego, California, USA; 2001.
Tan Yi-Hong, Lü Kevin, Lin Ya-Ping. Organisation and management of shared documents in super-peer networks based semantic hierarchical cluster trees. Peer-to-Peer Networking and Applications 2012;5(3):292–308, http://dx.doi.org/10.1007/s12083-012-0123-0.
Tan Yunsong. A multi-agent approach for P2P based resource discovery in grids. In: Paper presented at the proceedings of the 2009 international joint conference on artificial intelligence; 2009.
Tan Yunsong, Han Jianjun, Wu Yuntao. A multi-agent based efficient resource discovery mechanism for grid systems. Journal of Computational Information Systems 2010;6(11):3623–31.
Tangpongprasit Sanya, Katagiri Takahiro, Kise Kenji, Honda Hiroki, Yuba Toshitsugu. A time-to-live based reservation algorithm on fully decentralized resource discovery in Grid computing. Parallel Computing 2005;31(6):529–43.
Toninelli Alessandra, Corradi Antonio, Montanari Rebecca. Semantic-based discovery to support mobile context-aware service access. Computer Communications 2008;31(5):935–49.
Trunfio P, Talia D, Papadakis H, Fragopoulou P, Mordacchini M, Pennanen M, et al. Peer-to-peer resource discovery in grids: models and systems. Future Generation Computer Systems 2007;23(7):864–78.
Vanderster Daniel C, Dimopoulos Nikitas J, Parra-Hernandez Rafael, Sobie Randall J. Resource allocation on computational grids using a utility model and the knapsack problem. Future Generation Computer Systems 2009;25(1):35–50.
Wang Shaowen, Liu Yan, Wilkins-Diehr Nancy, Martin Stuart. SimpleGrid toolkit: Enabling geosciences gateways to cyberinfrastructure. Computers & Geosciences 2009;35(12):2283–94.
Waterhouse Steve. {JXTA Search}: distributed search for distributed networks; 2001. Retrieved from ⟨http://gecko.cs.purdue.edu/GNUnet/papers/JXTAsearch.pdf⟩.
Xiao-Hua Song, Yuan-Da Cao, He-Qing Huang. The distributed spanning-tree based service discovery in grid environment. In: Paper presented at the 2006 international conference on machine learning and cybernetics; 13–16 August 2006.
Yu Jia, Venugopal Srikumar, Buyya Rajkumar. Grid market directory: a web services based grid service publication directory. CoRR, cs.DC/0302006; 2003.
Yulan Yin, Huanqing Cui, Xin Chen. The grid resource discovery method based on hierarchical model. Information Technology Journal 2007;6(7):1090–4, http://dx.doi.org/10.3923/itj.2007.1090.1094.
Zhang Xuehai, Freschl Jeffrey L, Schopf Jennifer M. Scalability analysis of three monitoring and information systems: MDS2, R-GMA, and Hawkeye. Journal of Parallel and Distributed Computing 2007;67(8):883–902.
Zhao Chen, Yu Jian, Chai Bencheng. A study on mobile agent based resource management in grid. In: Apolloni B, Howlett R, Jain L, editors. Knowledge-based intelligent information and engineering systems, vol. 4693. Berlin, Heidelberg: Springer; 2007. p. 565–70.
Zhu Cheng, Liu Zhong, Zhang Weiming, Xiao Weidong, Xu Zhenning, Yang Dongsheng. Decentralized grid resource discovery based on resource information community. Journal of Grid Computing 2004;2(3):261–77, http://dx.doi.org/10.1007/s10723-004-2810-4.