

Resource Discovery for Distributed Computing Systems: A Comprehensive Survey

Javad Zarrin, Rui L. Aguiar, João Paulo Barraca

Javad Zarrin {javad@av.it.pt} Instituto de Telecomunicações - Aveiro. Rui L. Aguiar {ruilaa@ua.pt} Universidade de Aveiro, Portugal. João Paulo Barraca{jpbarraca@ua.pt} Universidade de Aveiro, Portugal.

Abstract

Large-scale distributed computing environments provide a vast amount of heterogeneous computing resources from different sources for resource sharing and distributed computing. Discovering appropriate resources in such environments is a challenge which involves several different subjects. In this paper, we provide an investigation on the current state of resource discovery protocols, mechanisms, and platforms for large-scale distributed environments, focusing on the design aspects. We classify all related aspects, general steps, and requirements to construct a novel resource discovery solution in three categories consisting of structures, methods, and issues. Accordingly, we review the literature, analyzing various aspects for each category.

Keywords: distributed systems, resource sharing, resource description, P2P, Grid computing, HPC

Acronyms

ACO Ant Colony Optimization.

AI Artificial Intelligence.

ANN Artificial Neuron Network.

ASL Average Search Length.

AVL Georgy Adelson-Velsky and Evgenii Landis.

BCA Bee Colony Algorithm.

BF Bloom Filter.

BFS Breadth First Search.

BOINC Berkeley Open Infrastructure for Network Computing.

Cell BE Cell Broadband Engine.

CP Client Proxy.

CPU Central Processing Unit.

DAML DARPA's Agent Markup Language.

DFS Depth First Search.

DHT Distributed Hash Table.

DIS DHT Information Service.

DNS Domain Name System.

DOS Distributed Operation System.

DoS Denial of Service.

FALKON Fast and Light-Weight Task Execution Framework.

FPGA Field Programmable Gate Array.

GIIS Grid Index Information Service.

GIS Grid Information Service.

GMD Grid Market Directory.

GPU Graphics Processing Unit.

GRIS Grid Resource Information Service.

HPC High Performance Computing.

HTC High Throughput Computing.

LDCE Large-scale Distributed Computing Environment.

MAS Multi Agent System.

MBFS Modified Breadth First Search.

MDS Monitoring and Discovery System.

MTC Many Task Computing.

NoC Network on Chip.

OIL Ontology Inference Layer.

P2P Peer-to-Peer.

PA Primary Attribute.

QoS Quality of Service.

RAM Random Access Memory.

RCaT Resource Category Tree.

RD Resource Discovery.

RDF Resource Description Framework.

SIMD Single Instruction, Multiple Data.

SLA Service Level Agreements.

SOA Service Oriented Architecture.

SOC Self Organized Cloud.

SON Semantic Overlay Network.

SP Server Proxy.

TORQUE Tera-scale Open-source Resource and Queue Management.

TTL Time To Live.

UDDI Universal Description, Discovery and Integration.

VEE Virtual Execution Environment.

VEEH Virtual Execution Environment Host.

VEEM Virtual Execution Environment Management System.

VM Virtual Machine.

VO Virtual Organization.

WRMS Workload and Resource Management System.

WSDL Web Services Description Language.

XDR External Data Representation.

XML Extensible Markup Language.

1. Introduction

In recent years, large-scale heterogeneous computing infrastructures such as Grids [1], Clusters [2], Clouds [1] or even the simultaneous combination of multiple platforms (e.g., Clusters, Grids, and Clouds used concurrently) have become increasingly widespread. This happened due to the development of manycore technologies and associated advances in high-performance and distributed computing. Despite the fact that these platforms provide distinct computing environments, they have a fundamental common key property; the ability to share resources/services belonging to different administrative domains among all the entities distributed across the whole system. Resource sharing produces significant benefits exploring the idle cycles of potentially available resources over the system by integrating, leveraging, and utilizing the potential of these myriads of individual available resources to achieve higher computing capabilities. The novel challenges arise from the fact that it would not be feasible to statically maintain the global knowledge of all the available distributed sharing resources for the existing entities in a dynamic Large-scale Distributed Computing Environment (LDCE). Indeed, one of the most challenging essential issues in such environments is the problem of resource discovery where each entity in the system has to be able to potentially explore and involve the desired resources to attain the relegated computations and services.

Resource discovery encompasses locating, retrieving, and advertising resource information, being an essential building block for fully exploiting all distributed resources in the system. The purpose of resource discovery in a LDCE is to enable the adaptation of the application's demands to the resources potentials by discovering and finding resources with a deep understanding of the resource specifications according to application's resource requirements. However, considering the nature of LDCE, any approach for resource discovery in such environment needs to be qualified in some challenging issues such as scalability, efficiency, heterogeneity, and reliability. As discussed in this paper, there are considerable amounts of research works done in the area of resource discovery which study these challenges in different aspects pursuing assorted objectives. In this paper, we provide a survey of the discovery protocols and their relevant aspects specific for large-scale computing. We describe all the potential related concepts and terminologies. Accordingly, we provide a taxonomy to categorize the various discovery systems and techniques while we analyze and compare different approaches. We also limit our study to discovery solutions which are applicable to be employed in LDCE. Although this paper does not highlight discovery aspects related to security, synchronization, and information summarization due to space constraints.

There are several surveys on resource discovery protocols in the current literature [3–9]. However, most of these works are limited, as they do not provide a comprehensive review of discovery approaches with respect to all key potential aspects of resource discovery in distributed systems. For example, the works [3, 4, 6] only discuss discovery approaches for Grid systems while the works [5, 7–10] only investigate approaches for

Peer-to-Peer (P2P) systems. Works such as [11–13] are much more limited in scope and dimension, and as such do not provide a complete view covering major aspects of discovery in distributed systems. Furthermore, comparing to this paper, none of these related works available in the literature discussed the evaluation aspects of discovery protocols. Moreover, this paper presents a fundamentally different approach to reviewing the current literature, introducing a practical research approach which can efficiently help researchers to design and develop novel discovery approaches in the area of distributed computing systems. Contrary to the other studies mentioned above we specify a set of major discovery relevant aspects first, and then we classify and discuss discovery approaches from the perspective of each particular aspect.

In this paper, we define a resource as any source of supply [12]. In the specific case of large-scale computing, we limit the definition of resource to computing resources that mean any element which is directly involved in computing process such as Central Processing Unit (CPU)-capabilities, Random Access Memory (RAM), disk space, communication and network capabilities, etc.

2. Relevant Discovery Aspects

For proposing a resource discovery solution, there are several fundamental open issues and involved elements which have significant impacts on the final discovery behavior, operation, and functionality. These may also affect the way that a resource discovery approach can deal with particular challenging requirements and objectives. In this paper, we have categorized and explained all the major general concepts, structures, techniques, and functionalities that shape the critical aspects of every resource discovery models in three classes, as described below.

1. **Underlying Aspects** contain all the things that should be prepared and configured before discovery usage (e.g., stating a query). These aspects are more relevant to the underlying computing environment, network topology data infrastructure, representation of resources (using resource description models/languages), and resource information distribution (in terms of architecture).
2. **Design Aspects** include the current major possible relevant techniques, strategies, and methods that can be used to guide queries between distributed entities for locating and discovering resources. Search algorithms, mechanisms for packet propagation (i.e., communication models for distributing query/resource information across the network), querying strategies, resource information delivery, data synchronization between distributed resource providers, and information summarization techniques (for compact querying with minimizing the size of query data transferred) are examples of such elements.
3. **Evaluation Aspects** cover all the concepts and terms that express, demonstrate or evaluate the expected or desired achievements of the resource discovery procedure. Depending on the specific environment, applications and objectives

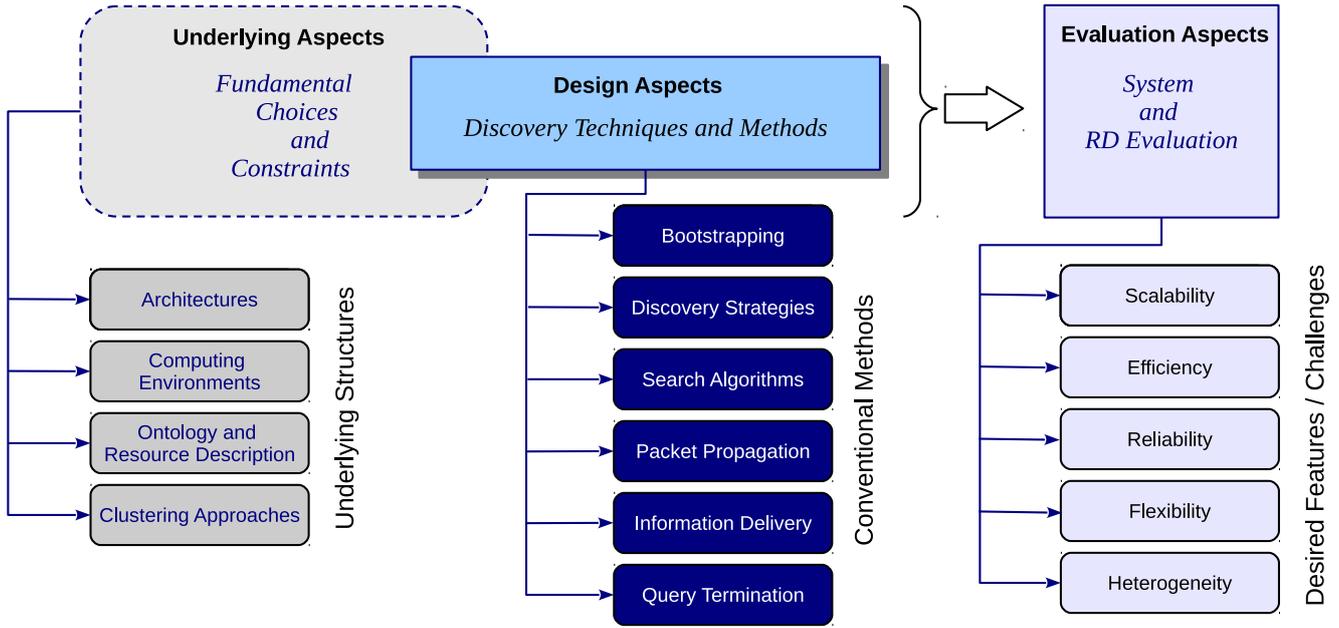


Figure 1: Classification of the relevant resource discovery aspects.

that a discovery protocol has been designed for, the number of certain functionalities, features and performance factors can become critically important to attain. For example, for the resource discovery in wireless sensor networks, energy efficiency is one of the critical performance factors while in the area of large-scale computing, scalability has the key role.

Figure 1 illustrates the relationships between the aspects mentioned above. The figure also demonstrates a research path to design new resource discovery protocols for distributed systems. It can be depicted as a very coarse-grained workflow (tutorial) which provides an overall research map to researchers that aim to propose discovery approaches. In the first step, underlying aspects must be investigated. Accordingly, it is needed to introduce strategies to build **underlying structures**. In the next step, there are a set of design aspects which must be taken into account in order to propose a novel discovery approach. This includes a detailed review of current **conventional methods** in the literature, addressing different design aspects. Finally, in the last step, after developing a discovery proposal, this must be assessed accordingly to several evaluation aspects. Researchers pursue different objectives for proposing a discovery approach. Therefore, evaluation aspects emphasize on various **desired features**, dealing with different **challenges**.

In the continuation of this paper, we discuss and explain the elements mentioned above which play a vital role in designing and operating of resource discovery protocols.

3. Underlying Aspects

In this section, we discuss resource discovery approaches in the literature from the perspective of the more fundamental underlying aspects. The underlying aspects are related to any

fundamental basis (such as architectures, structures, and environments) which provide the ground (basic infrastructure) for the act of resource discovery. We first elaborate on the general architectures of distributed systems including advantages and disadvantages, and how these can fit with the resource discovery requirements. Accordingly, we explore the resource discovery literature concerning characteristics of different specific, real-world, computing systems/environments, based on various distributed architectures. Furthermore, we discuss ontology and resource description approaches which are required to describe, abstract, and organize information on hardware resources (and their capabilities), presented in a computing environment. This information and abstractions, in turn, can be used to construct virtual overlays on top of real physical systems through applying clustering approaches. An overlay (or a virtual overlay network) is a structure which provides underlying data and virtual topology to facilitate building discovery components and services. The section will end by discussing common approaches for clustering and overlay construction.

3.1. Architectures For Resource Discovery

In the distributed system, before designing a resource discovery model, it is necessary to examine the general distribution options in environments and clarify an architecture depending on the application, adaptation to the target computing environment and a set of other critical involved parameters. From the state-of-the-art [3, 4, 14–18], we can categorize the discovery architectures in four distinguishable groups which are centralized, decentralized, hybrid (e.g., decentralized-tree and decentralized mesh), and hierarchical.

In the centralized case (e.g., [19–23]) (Figure 2), the resource information about all other nodes and instances is located at a central point, that can be reached by all service or resource requester instances in the environment. All the resource providers

periodically update and register their dynamic or static information in the central repository. The central information service is the only entity in the system who can process the queries, initiated by resource requesters, for matching to the resources available. The centralized discovery has the advantage that the information source is always known and that reallocation of services/resources can be more easily propagated. At the same time, the central information service becomes a bottleneck and a single point of failure. In fact, fully centralized systems often do not scale well.

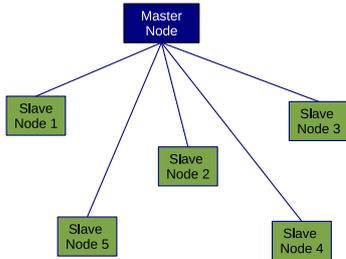


Figure 2: Centralized architecture.

As opposed to the centralized architecture, in the decentralized case (e.g., [14, 24, 25]) (Figure 3), all (or, at least, several) nodes have to host at least the base information and capabilities related to query processing and management. Thus, requesters do not need to ask a central instance for query analysis and receiving information about the desired resources every time. Accordingly, the system becomes more failure resilient and in particular much more scalable. However, depending on the use case, the degree of distribution can quickly lead to a management communication overhead if all nodes have to share the same information from different endpoints. In such situation, the decentralized approach leads to an all-to-all communication (e.g., query flooding) across the whole network. But, it might be possible to reduce the overhead from all-to-all communication (broadcast) to one-to-many (multicast or anycast) or one-to-one (unicast) communication by using mechanisms such as selective search and multicasting (refer to Section 4.3).

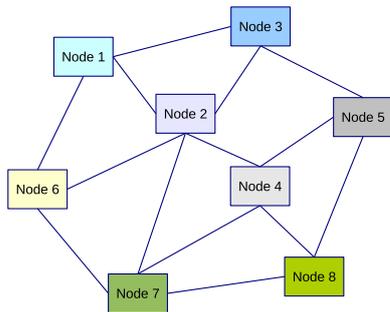


Figure 3: Decentralized architecture.

Nevertheless, it is always harder to keep track of potential query-resource reallocations (caching), as the resource information (e.g., resource location) either has to be distributed to all possible communication points, or some mechanisms for routing and updating have to be introduced.

Assuming that we have n nodes in a centralized system, the minimum/maximum query cost is one message (regardless of

returning messages) since a requester can simply send a query to a central master-node which can process the query and return a proper response by analyzing all information of resources, collected in the master-node. The query cost might be two messages if the master-node only processes static specifications of resources. In such a case, the master-node may send a query to the corresponding resource node for checking dynamic information on the site. The centralized system also has maintenance cost since resource information might be changed during runtime. Assuming that all nodes periodically update their information on the master-node, the maximum maintenance cost per interval is $n - 1$ messages since all nodes (except the master-node) must send an update message (including changes of resource information) to the master-node in each interval of time. In fact, centralized architectures provide efficiency concerning the query cost while they may not be efficient in terms of the maintenance cost.

In (fully) decentralized models, there might not be a maintenance cost, but the query cost is proportionally higher than centralized architectures. The minimum query cost is one message if we assume that the query will be resolved by the first visited node. The maximum query cost is n messages if we assume that all nodes in the system will be visited exactly once. Theoretically, the worst situation can happen when resolving a query requires all-to-all communication among all nodes in the system, resulting in a maximum query cost of $n(n - 1)$ messages. However, this may not happen in real scenarios since even a single node in the system can resolve any query by broadcasting $n - 1$ messages to all nodes in the system. The query response will be deterministic since all nodes are explored by broadcast messages. Further discussion on this topic is presented in Section 4.3.2 for different specific search methods.

We can also compare centralized models to decentralized architectures in terms of deterministic results for queries. A query is deterministic if a decision can be guaranteed as the query result. In a centralized structure, the master node can provide a deterministic response whether the resource required for a given query is available in the system or not. But, in a decentralized model, depending on the case and the given query, a deterministic response may not be attainable unless all nodes are visited systematically or algorithmically. Visiting all nodes (or potential nodes) in the system increases the query cost. Furthermore, in decentralized models, if an adequate search method is not conducted, queries can be propagated indefinitely by revisiting redundant nodes (loops). This means that in decentralized models we may not receive a deterministic query response or if achieved it would be costly (in terms of query cost/overhead). A common strategy to overcome such issues is to use query termination techniques (refer to Section 4.6).

The hybrid models benefit from a combination of features of both centralized and decentralized architectures. Decentralized-tree (e.g., [26, 27]) (Figure 4) and decentralized-mesh (e.g., [28]) are types of hybrid architecture which have been designed to aggregate the advantages of the traditional architectures while reducing their shortcomings. As a practical example of this kind, we can mention the resource discovery in a manycore environment, where the individual resource description sources can

encapsulate multiple levels of aggregated resources. This higher level information can either integrate all lower-level resource descriptions or provide a more holistic view of the information by using abstraction. Either choice affects the communication strategy as more information implies a higher amount of data (and associated maintenance), while more abstract data implies that additional queries may have to be executed for acquiring details that are potentially required.

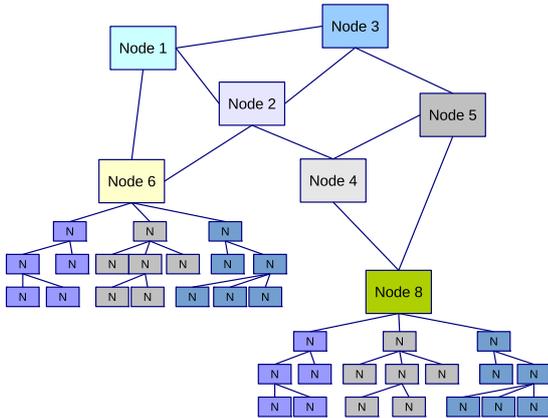


Figure 4: Decentralized-Tree architecture.

The hierarchical architecture (Figure 5) through which the relevant information is propagated in a hierarchical fashion, i.e., through locally connected layers that each, in turn, is connected to lower layers. This way, the information does not need to be propagated throughout the whole network, and the communication overhead is restricted to the substructure. However, at the same time, propagating the layers adds delay due to the messaging route so that lower levels will get access to the information later than higher levels.

A hierarchical structure can be established either statically or dynamically. For static hierarchies [29], the architecture can be configured by manually defining layers, members of each layer and relationships between layers/members. For dynamic hierarchies [30], the structure can be configured dynamically through self-organization of nodes in tree-like structures. Hierarchical models generally consist of three layers for processing resource information. These layers include physical resource, logical resource information and index information (top index or child-top indexes). The physical resource layer is at the lowest level containing real physical hardware resources of the system, connected through network links. Each logical resource (i.e., resource or resource node) represents specifications/characteristics of a physical resource in the second layer. Resources are connected through a star topology with a central super-node, positioned in the third layer (index information layer). Each group of resources with a super-node may be referred to as a Virtual Organization (VO) (e.g., in Grid computing, refer to Section 3.2.1). A super-node maintains all information of VO members (resources) through index tables (e.g., MasterTableViews for top-index super-nodes, GridTableView for child-index super-nodes, and DetailTables for resources). The super-nodes, in turn, can be connected through a ring topology.

All nodes in a hierarchical system must locally maintain a set

of information which are necessary for specifying their positions in the hierarchy. For example, a member node (resource) may contain information such as its detailed specifications, its role in the hierarchy (which can be "leaf-node" for member nodes) and the address of a super-node (its super-node ID). Similarly, each node with a "super-node" role may include indexed information of its leaf-node members as well as their leaf-node IDs. The initial allocation of these information can be done statically or dynamically, as we mentioned earlier. The initial information can also be changed dynamically during runtime.

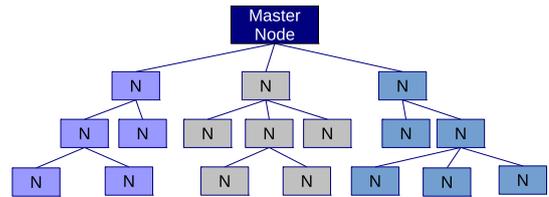


Figure 5: Hierarchical architecture.

The generic hierarchical architecture, mentioned above, can also be extended to support further layers by defining two types of super-nodes: child-index and top-index super-nodes, using the same star topology for both. A top-index provides aggregating information about a set of child-index nodes, and a child-index includes indexing information regarding a set of resources. The updating and maintenance of information in the index tables can be performed through periodical updating of state information (either by super-nodes or nodes). Furthermore, depending on the situation, the communication between a super-node and its member nodes can be done through unicast (e.g., for updating the state of a resource in the index-table by a resource-node) or multicast (e.g., for querying multiple resource-nodes by a super-node), where a multicast group includes all members of a super-node.

We must note that the hierarchical model can also be considered as a hybrid architecture since it combines aspects of both centralized approaches (it has a single head/master) and decentralized approaches (it requires communications with multiple levels of nodes, positioned in a hierarchy, to answer a query/request). However, in this survey, we consider the hierarchical model as a separated architecture since it is a well-known structure which is widely used in many works (e.g., see Section 3.2.1-B), in the area of distributed computing systems.

3.2. Computing Environments

The computing environment has a large impact on the architecture, operation, implementation methods, and the performance of resource discovery protocols. In this section, we describe and classify current resource discovery approaches for the most important (and more conventional) large-scale computing environments including Grid, P2P, Cloud, and Cluster (see Figure 6).

3.2.1. Grid Systems

Grid computing [1, 2] generally contains plenty of heterogeneous resources which potentially are loosely coupled and

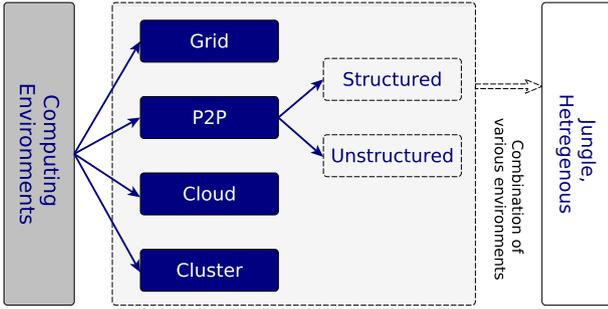


Figure 6: Large-scale computing environments.

geographically distributed. This “large group of resources” (devices, data, applications, services, storage, sensors, computational clusters, parallel supercomputers and any other sort of computing or communicating resources) act together as a single super-powerful computing system to perform large tasks. The Grid has originally been designed to efficiently map the user’s jobs to the most appropriate free resources. For processing user’s jobs, it is essential to get resource information in a minimum time. There are various types of Grids depending on their specific objective and focus such as data, application, service, knowledge, interaction, utility, and computing Grid. In this paper, we mainly address on computing Grids.

Resource discovery in Grid systems aids in resource management and application scheduling and generally suffers from two major challenges: efficiency and complexity. First, the majority of the existing Grid Information Services (GISs) [31] (like Monitoring and Discovery System (MDS) [32]) will organize their resources based on non-adaptive overlays where the categorization of resource characteristics and information doesn’t play any role in the overlay construction, being dependent on the virtual organization and administrative parameters. In fact, a non-adaptive (non-compatible) overlay does not benefit from the resource description approach which is used to describe Grid resources for the purpose of clustering and overlay construction. The incompatibility between the overlay clustering mechanism and the resource description model (in Grids) leads to inefficient query forwarding across the whole system while a resource-aware overlay clustering could reduce the discovery overhead significantly by limiting the query traversing area. The second challenge is providing the capability for Grid discovery to perform complex querying. Grid job schedulers are responsible for allocating user’s tasks to the most appropriate resources. Basic querying techniques like exact-matching are not enough to precisely map the job’s requirements to the resources specifications. The drawback is that the exact matching heavily depends on the exactness of the query’s conditions and it requires to specify the exact job’s requirements, which might be not feasible in practice. In other words, exact matching strategies are not suitable for approximate matching when the exact demands of the jobs are not clear. Moreover, exact matching ignores the resources which have specifications overqualified for the job’s requirements. This leads to underutilization of the system’s resources, causing the overqualified resources to be idle when exact matching fails. Thus, leveraging multiple complex querying techniques such

as keyword search, range querying, partial matching, multidimensional and resource graph querying becomes a necessity to perform adequate job/resource matching.

Resources in Grid are defined as a set of attributes. A query simply describes the specific desired values for some of the attributes. A simple example of a query is like “processor frequency=1.7 GHz” where the discovery process locates all the processing resources in the system which have exactly a 1.7 GHz processor. However, complex range querying like “2.1 GHz > processor frequency > 1.7 GHz” or multidimensional querying like “A1=V1, A2=V2, V4 > A3 > V3” generally are not supported in Grid environments. We will discuss complex querying issues in detail later.

Multiple Grid-based resource discovery approaches have been proposed to enable a GIS. They can be categorized in centralized, hierarchical, decentralized and hybrid systems according to their main approaches to the discovery problem. Table 1 presents examples of resource discovery in Grid systems.

A) Centralized-Grid: Centralized model is the simplest approach for creating an information service, where it is constructed by establishing a centralized directory agent. The major advantage of this solution is the simplicity of finding all resource information on the central server, making the resource discovery latency low, and data coherence high. However, these approaches suffer from sub-optimal scalability and lower fault tolerance, mostly due to the centralized nature of the directories, as discussed previously in Section 3.1. Nimrod-G [33] and Condor-G [34] are the examples of Grid super-schedulers where they have employed a centralized Grid information services such as R-GMA[35–37], Hawkeye[37, 38] and Grid Market Directory (GMD) [39] to index their resource information.

B) Hierarchical-Grid: Another approach for discovery in Grids relies on hierarchically organized servers. In MDS [32, 41], the top index server answers requests either directly or by dispatching them to its child index servers. This approach has limited scalability, as requests trickle through the root server, which can easily become a bottleneck and consequently suffer from fault tolerance issues. Indeed, the loss of a node in the higher level of the architecture causes the loss of an entire subtree. Ganglia [45], MDS-3 [47], and MDS-4 [43, 44] are other examples of hierarchically designed GIS.

C) Decentralized-Grid: In decentralized Grid discovery, the information requester and the information provider (or resource broker) are two discovery agents which interact with each other in a self-organizing and self-adapting manner. The broker agent contains a resource information database that must be discovered from other Grid peers, to provide a global knowledge view to the local information requesters. The concept of decentralized discovery systems in Grid is related to the idea of peer to peer computing wherein there is no central server in the system. We discuss further details of these approaches, separately in Section 3.2.3, which present discovery approaches benefiting from a mixture of P2P and Grid.

D) Hybrid-Grid: Hybrid approaches have been specially designed to provide a high level of scalability and fault tolerance, which is required in large-scale environments. They are proposed to enhance and extend the natural capabilities of

Table 1: Examples of resource discovery in Grid-based systems (DS: Discovery System).

DS	Base	Mechanism
GMD [40]	Centralized-Grid	Provides a web-based approach for managing resources in Grid systems. It consists of two main components: a portal manager and query web service. Grid users can share and register their resources in Extensible Markup Language (XML) format manually in the central resource information repository, using the portal manager. Clients can search for their required resources in the central database using the query web service. The central node is the only peer in the system which can gather and store resource information and perform the query processing. Therefore, it easily becomes a bottleneck. Like as any other centralized system it suffers from poor scalability. Furthermore, resource owners should manually update their resource information, as GMD doesn't support dynamicity in all aspects (e.g., dynamic attributes, dynamic topology). However, it supports some complex querying features such as multidimensional and range querying since it has a central database to store resource information which those kinds of queries easily can be supported.
MDS-2 [37, 41]	Hierarchical-Grid	Designed to enhance the primary version of MDS in order to be adapted in a hierarchical environment. It has two main components, a configurable Grid Resource Information Service (GRIS) and a directory component called Grid Index Information Service (GIIS). They are organized in a hierarchy where the higher nodes host GIIS and the lower nodes host GRIS. The information providers register and store all the resource information of their local resources such as static and dynamic attributes and their specific virtual organization policies in the aggregate directories in the higher level using GIIS. Consequently, resource providers periodically update their local resource status in their registered directories. If an information provider after a given interval fails to update the directory, GIIS assumes that the information provider is not available anymore and it removes the provider from the index list of validated GRIS.
MDS-3/4 [42-44]	Hierarchical-Grid (tree-style)	Proposed refinements of MDS-2. They follow a tree-style hierarchy within a VO to dictate node resource information dissemination. A virtual organization is a group of Grid peers that agree on some common resource sharing policies. Every VO designates a node that hosts all the local resource information. These representative nodes register themselves in the related directories where they are organized in the hierarchy, based on specific resource information. In MDS sometimes it might happen that a low-level node begins to stop updating and sending resource information to the GIIS. Thus, the MDS search (in both centralized and hierarchical manner) could not be deterministic which means if the query answer is available in the system, it is not guaranteed to see the answer in the query result. In other words, the system could not provide a reliable resource information to the clients. Furthermore, as the size of the VO and the query rate increase, MDS has scalability limits due to the high network traffic near the root node.
Ganglia [45]	Hierarchical-Grid	Proposed a distributed monitoring system for high-performance computing systems such as clusters and Grids. The resources are organized in a set of federated clusters. In each cluster, a representative node reports the status of resources within the cluster to the federated monitoring system. It leverages various technologies such as XML for resource description, External Data Representation (XDR) for data packing and transferring resource information, and Round-Robin Database tool (RRDtool) for data structuring and storing information. It has tried to achieve a scalable solution by reducing network overhead per node and increase the level of concurrency, however, like MDS-3, it still suffers from similar problems to the centralized systems such as single point of failure and limited scalability.
RCaT [46]	Hybrid-Grid (decentralized-tree)	According to the priorities of the resources characteristics, Resource Category Tree (RCaT) organizes and categorizes the resources in form of Georgy Adelson-Velsky and Evgenii Landis (AVL)'s trees (i.e., self-balancing binary search trees) where each node in the tree is responsible for managing the resources for the values of a specific attribute within a range, instead of a single attribute value. The tree is organized based on a self-balanced structure from top to down where each level categorizes the values ranges for a particular attribute. Each tree root must have an assigned Primary Attribute (PA) which is the most important attribute that can define the resource's capability. The PA allocation might also be relevant to the application resource requirements. For example, in the case of an intensive computing parallel application, the number of cores in the processor (static attribute) or the current load of the processor (dynamic attribute) might be a potential PA. Each node in the tree only stores information about its connection links to the child nodes, the parent node and also the range values which the node is responsible for them. Therefore, in RCaT, it is not necessary to store large amounts of information in the higher nodes, achieving better scalability. However, the maintenance cost to create different dynamic trees based on different attribute priorities is high, and it has a significant impact on the overall system efficiency. In comparison to the traditional trees and hierarchical systems, the distribution of the query loads in the RCaT's trees efficiently has been balanced and also the Average Search Length (ASL) or the number of involved nodes to process a query has been decreased. In overall, we can say that RCaT has improved and enhanced the efficiency to traverse queries in tree style structure, but it remains unsolved the number of critical issues related to scalability, heterogeneity, dynamicity and even complex querying.

the centralized and hierarchical models (such as flexible querying and reliability, concerning deterministic search and discovery correctness) to a wider set of desired features and capabilities in different aspects. A hybrid model can be achieved by aggregating a central/hierarchical approach to the advantages of the fully decentralized models (such as scalability, reliability, self-organization, and efficiency). Decentralized-tree and decentralized-mesh are the sample architectures which have been used to deploy hybrid solutions like RCaT [46] for resource discovery in Grid.

3.2.2. Peer to Peer Systems

Peer to Peer computing systems has emerged as an alternative paradigm to construct large-scale distributed systems. The concept of P2P introduces many significant advantages in different aspects of resource discovery including scalability (due to collaborative resource sharing between peers), reliability (e.g., fault-tolerance) (due to the equality essence of peers), robustness (due to self-organization against peer or system failures). For resource discovery in P2P systems, it is more common for peers to create network overlays on the top of physical network topologies. These overlays might provide a specific structure to store and distribute resource information among peers, or the peers

can follow dynamical information maintenance and distribution behaviors in an unstructured ad-hoc fashion. Table 2 elaborates some recent examples of P2P resource discovery approaches which are based on different overlay architectures. In continuation of this section, we discuss the different approaches in P2P, which we organize in structured, unstructured and hybrid.

Structured Systems: Most P2P resource discovery systems depend on a structured architecture (e.g., ring, tree) where the system can be understood by certain nodes of which the resources information are fixed. Such systems are in general faster in discovering resources than unstructured Resource Discoverys (RDs), with a fixed and predictable time of resource discovery. These approaches can be extremely useful for searching resources using unique identification, but they fail when a search using partial matching is required. Moreover, they create additional overhead due to the management of the network architecture (by updating the system structure in the case of node failure or arrival).

There are two types of structured overlays: Distributed Hash Table (DHT) based systems (such as Chord [60], CAN [61], Pastry [62], Tapestry [63], P-Grid [64], and D-Grid [65]), and non-DHT based solutions (like Mercury [66]). However, most overlays of this type commonly make use of DHTs. A DHT

Table 2: Examples of resource discovery in P2P-based systems (DS: Discovery System, FoCs: Flocks of Condors, MaT: MatchTree, CyG: CycloidGrid, SkipC: SkipCluster).

DS	Base	Mechanism
MaT [48]	Hybrid P2P Overlay	Proposes a scalable and fault tolerant system by creating a self-organized tree for query distribution and result aggregation with a specific asymptotic latency increase pattern. It reduces the query latency and improves the system fault tolerance through redundant query topologies, sub-region queries, and a set of progressive timeout policies. It supports complex queries and guarantees query completeness.
Tripod [49]	Hybrid P2P Overlay	Proposes an inherent fault-tolerant system based on hybrid overlay network for enhancing the scalability and search efficiency of resources in highly dynamic large-scale heterogeneous environment. It is efficient to discover resources in proximity while it provides dynamicity in terms of dynamic resource attributes. Additionally, the network overhead cost for overlay construction and maintenance is very low.
CyG [50]	P2P overlay	Provides a two stages Quality of Service (QoS) and locality aware discovery algorithm for P2P based volunteer computing systems. In the first step, it discovers a set of resources based on the required quality of service and the current load of the peers. In the next phase, it selects the closest resource concerning communication delay (latency) between peers which is calculated using a network model based on queuing theory with consideration to background network traffic.
PIRD [51]	DHT-based	Focuses on multi-attribute querying and locality awareness. The system has been built based on a hierarchical P2P structure which uses a locality sensitive hashing to generate indices for the peers and resources in a DHT overlay. It leads the system to be able to discover resources geographically near to requesters. PIRD incorporates the Welch algorithm [52] to compress attribute information and then it weaves all attributes into a set of indices using the hash function. Thus, the system can effectively define and perform a multi-attribute query by sending a single query message to the network.
FoCs [53]	Pastry (DHT-based)	Combines the flocking of the Condor [54–56] pools with the Pastry mechanisms. It uses a self-organized Pastry overlay to index the distributed condor pools. The system is static and doesn't support dynamicity
SkipC [57]	Hierarchical P2P Overlay, Pastry (DHT-based), SkipNet [58], SkipGraph [59]	Designed to solve the problem of range querying and other kinds of proximity-aware related complex querying in classical DHTs. The system can support both exact matching and multidimensional range querying without storing extra information in the peers. It has two tiers of hierarchical architecture where in both tiers the peers are ordered and organized based on the sequence of their identifiers, considering the point that the semantically related peers are stored near each other without hashing their resource keys. The pointers to the super peers in the higher tier are stored in a new data structure called Triple Linked List (TLL) which are used to find the longest prefix for the query key in the remote clusters. In the lower tier, the routing table of each peer contains pointers with exponentially incremental distance.

is a distributed data structure to efficiently perform distributed storage and lookup of pairs (key, data) which provides a scalable, fault tolerant and fast locating of data when a key is given. Here we explain some of the well-known DHT based solutions in detail.

Chord organizes peers and resources in an m -bit, ring based, identifier space where m represents the total number of required bits to specify each fixed-length keys/identifiers and the maximum number of 2^m keys (representing resources or entities) can be identified in the interval of $[0, 2^m - 1]$ (a range of non-negative integers). We also must note that the value of m should be large enough to avoid the collision in hashing. The Chord ring is the basis for efficiently locating the peer that contains a particular data item (i.e., resource description or resource information). Both peers and resources are assigned m -bit identifiers and ordered based on their identifiers in a circle modulo 2^m (called Chord ring). The organization of peers is done by employing SHA1 (Secure Hash Standard), a consistent hashing function, which hashes the peer's location (IP address) and the key (e.g., definition of a resource by a keyword, summary of the resource information) respectively. Each key is assigned to the first peer clockwise from the key in the Chord ring (called successor peer of the key) whose identifier is equal to or pursues the identifier of the key. Predecessor also is the peer that takes a position right before the key in the ring. The distribution of the keys between peers is load balanced by roughly allocating all the peers an equal number of keys where each peer must maintain the correspondent data (resource description) for several resource keys in the form of (key, data) pairs.

The process of locating resources can be implemented on top of Chord overlay by allocating a key with each data (resource description) item and storing the key/data pairs at the correspondent peers. The process is started by issuing a query to the

system. Afterward, the query hashed to a specific key identifier by consistent hashing function, given a key, considering the point that each peer knows about its successor, the queries can be forwarded around the ring using the successor pointers until the successor peer who is the owner of the key is encountered. The successor peer of the key is responsible for storing the required resource information related to that key in the query. In order to enhance the efficiency of the algorithm and limit the number of explored peers to handle a query, each peer maintains an m -entries small finger table (see Figure 7), which contains the pointers to the potential successors for each range of keys.

A new peer can join the system by contacting a known peer already included in the Chord ring. The joining request can be responded by using the address of the potential successor and predecessor. The new peer will ask these peers to be added to the ring. Consequently, the number of certain keys which were previously assigned to the successor peer will be moved to the new peer. Similarly, when a peer voluntarily departs the system, it transfers its keys to its successor peers in order to maintain the resource availability of the system. Sometimes, it might happen that a successor of a peer suddenly leaves the system, in such a case each peer can substitute the ring with another successor in the list of alternative successors which already have been stored in each peer.

Sudden involuntary leave of peers can affect the resource availability because resources in the Chord system are not replicated, crosswise peers. Thus, the system can designate an alternative peer as the replacement for the departed peer. But there is no guarantee to find an equivalent resource in the system instead of the resource which already becomes unavailable due to its peer's departure. This will decrease the reliability of the Chord system to a certain amount. Moreover, to address the problem of frequent departures and joins of the peers in the system, Chord

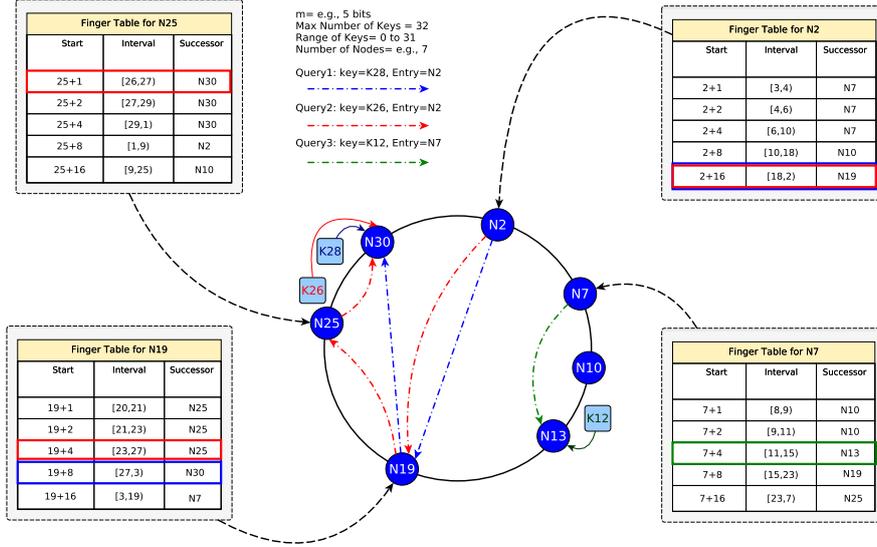


Figure 7: Examples of query processing in Chord.

uses a stabilization protocol which runs on each peer periodically to maintain the successors' information and finger tables updated. However, this is a costly process which affects the system efficiency by occupying the bandwidth to handle the maintenance loads. Chord distributes the resources among the peers in an appropriate load balanced fashion while it reduces the bandwidth cost of the query by avoiding flooding. The combination of these performance factors result that the system becomes extremely scalable, additionally exploiting the capability of the finger tables leads the Chord to facilitate faster and efficient querying.

However, the Chord DHT has some disadvantages, for example, due to using the hash values, it is only able to perform exact querying, while it can not resolve a reliable complex query (e.g., range, multidimensional and keyword search). Furthermore, the peers in Chord based resource discovery solutions (like pSearch [67]) do not have full autonomy to control their local resources. Thus, the Chord system could not be a purely decentralized system. Research works in [68–75] discuss some examples of improvements on Chord. Figure 7 demonstrates examples of query processing using Chord.

Pastry [62] and Tapestry [76] are two distributed lookup systems which are very similar to Chord. The difference is that in Chord, query forwarding is performed based on the numerical differences between the source and destination address while in Pastry and Tapestry the request forwarding is based on prefix and suffix based routing approaches respectively. Moreover, despite the similarity to the Chord, they provide support for locality-aware discovery, which ensures that peers and distances in the logical overlay network have a correlation with the physical nodes and network distances in the underlying layer. Therefore, passing a query along a logical connection to a close peer in overly does not lead to a long distance query traveling in the underlying network.

Pastry creates an m -bit (generally $m=128$) identifier space in the range of $[0, 2^m - 1]$ by using a cryptographic hash function. The IP address or the public key of each peer is hashed to a

unique node-id. Consequently, the peers and resources must have their assigned m -bit node-ids and key identifiers respectively and like other DHTs the keys should be distributed among the peers in a load balanced manner. The peers are organized and arranged in ascending order of node-ids in a ring. By having a key for lookup, the system routes the query to the peer whose node-id numerically is closest to the m -bit hash value of the key. For implementing the lookup procedure and query forwarding on the top of Pastry ring, each Pastry node is required to store three type of data items, routing table, neighbors table and leaf set nodes. The routing table of each peer stores the pointers to the (usually long distance) peers in other prefix realms which share different lengths of prefix (first i digits) with the node-id of the peer within the system.

In Pastry, ring consists of N nodes with definition of b bits length (for each digit) and the identifier $base=2^b$. The routing table of each node has $2^b - 1$ columns and $\log_{2^b}(N)$ rows. The entries at row i refer to peers that share the first i digits of the prefix with the node-id while the maximum entries for each row are $base - 1$. The $i + 1$ th digit of the entry in cell (i, j) must be equal to the column number of j . In other words the entry of each cell (i, j) of the routing table must contains the topologically closest node with prefix length i and $digit(i + 1)=j$. The leaf set nodes contain the pointers to the l nodes which their node-ids are numerically the closest nodes ($l/2$ nodes with a numerically closest larger node-ids and $l/2$ with closest smaller node-ids). The neighbors' table stores the node-ids of the peers which are topologically nearest (e.g., concerning latency or network hops) to the current node. The neighboring information can be achieved by caching of nearby candidates for routing table during the construction of the routing table. The pastry algorithm to find a key k in the system starts by submitting a query to node p which is already included in the ring. Node p checks its local leaf set nodes to find if a match for the key k is within the leaf set if so, the query is resolved by forwarding the request to the matched node which already owns the required resource of the query k . Otherwise, referring to the information provided by

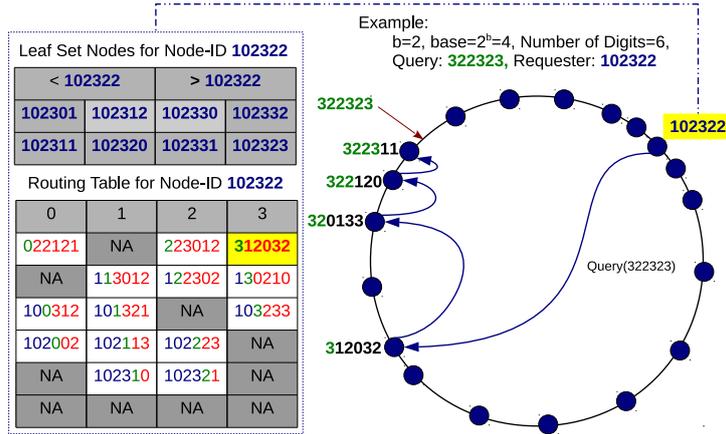


Figure 8: Routing example in Pastry: Query with the key 322323 arrives at the node 322311 which has the closest node-id. The routing algorithm corrects one digit at each step and then uses "leaf-set" to locate node with closest node-id to target.

the routing table, Pastry forwards the query messages to a node which has one more matching digit in the common prefix (see Figure 8).

In the rare case when node p can not determine another node in the routing table (a node which provides a longer length of matching prefix), the query is forwarded to any node that is closer to the key than the current node-id within the merged set of the routing table. The merged set includes neighborhood set and the leaf set. Research works in [77–82] are some examples of the improvement works based on Pastry.

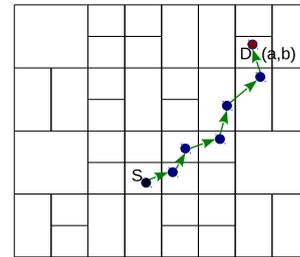


Figure 10: Sample routing path from S to D in a 2-Dimensional CAN. Using the greedy routing strategy, the query message, in each intermediate node, is routed to a neighboring node that is positioned closer to the desired location.

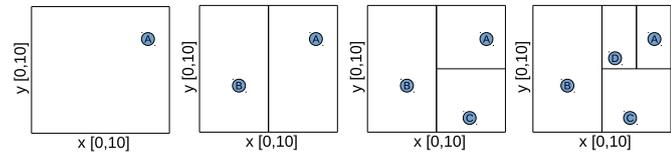


Figure 9: Example of partitioning in 2-Dimensional CAN. Each node is assigned a unique zone in the M -Dimensional coordinate space. A new node is usually joined by identifying and splitting a zone in the coordinate space that can be split.

The scalable P2P models such as Chord, Pastry, Tapestry and CAN (see Figs 9 and 10), have two general disadvantages. First, they do not provide fully local control over data in each peer which reduce the overall decentralization, autonomy, and flexibility of the system. Secondly, they do not guarantee that the routing paths will remain permanently within a constant administrative domain, which reduces the reliability and increases the maintenance cost of the system. There is some research works in P2P such as SkipNet, SkipGraph, P-Ring [83, 84] and Online Balancing [85] which concentrate on solving above issues.

Overall, we can conclude that DHT based systems build efficient query execution structures that are well suited to deal with the scalability and efficiency issues for discovering resources in LDCE. However, they come with some shortcomings in different aspects such as semantic querying, dynamicity, heterogeneity end topology mismatching. Several research works are focusing on improving the capabilities of the DHTs along the line of the problems mentioned above. In the following, we explain the drawbacks and some of the potential solutions found in the literature.

DHTs do not support semantic querying and keyword search. The lack of semantically-rich resource descriptions and capability to perform flexible/complex querying in DHT limited the discovery efficiency, especially in the environment where the resource requesters are not able to precisely clarify their resource requirements and conditions. For example, a requester might be interested in locating a certain number of vector processors (e.g., Single Instruction, Multiple Data (SIMD) processing cores) in an attribute-based DHT system. Assuming that the resources are described based on a constant number of attribute values, the query might not be successful even if the qualified resources are present in the system. The reason is that simple exact query results require the query itself to be described precisely by the resource requester, and the resource requester does not want to limit its potential resource options, while the requester doesn't know about the available resources in the network. Semantic-based querying is one way to overcome this problem by organizing peers/clusters based on their contents (i.e., based on the conceptual similarity of the resources) to enhance content search.

ERGOT [86] presents an approach to overcome the problem of semantic querying in DHTs by leveraging the DHT mechanisms to build a Semantic Overlay Network (SON) [87]. DHTs are scalable and fault tolerant. They also guarantee efficient lookup at an exactly predictable cost while they are semantic-free. On the other hand, SONs are flexible enough to perform semantic driven querying, enabling them to go beyond exact matching (but are less scalable than DHTs). It has to be taken into account

that the performance of SONs highly depends on the way that semantic links are created, and the communication cost to create the semantic links. ERGOT enables semantic-based resource discovery in distributed infrastructures such as Clouds and Grids. It employs a sort of semantic annotations (required to define the notion of similarity) for enhancing and enriching the description of resources/services and queries by providing two strategies:

- First, peers can construct the semantic links during the normal interaction of peers in DHT by recognizing the peers with similar content. Thus, the semantic links (links between peers with similar content in SON) can be created without any extra cost. In other words, resource providers advertise their resources in the DHT according to their annotations which result in building SON among the resource providers which provides similar resources.

- Second, the system can use the advantage of DHTs to perform exact matching while it can resolve semantic queries based on resource match-making by measuring the similarity, between query's resource requests and resource descriptions.

Similarly, DHT Information Service (DIS) [88] proposes a scalable DHT and ontology-based information service for Grids to speed-up querying and enhance query precision and integrality, by aggregating DHTs and semantic-based query techniques. DIS is concerned in the dynamic attitudes of the resources (e.g., frequent joining and leaving of resources) in virtual organizations since it requires robust self-organization capability of the system to maintain the system structure. Additionally, in the case of creating key space for the DHTs, a large-scale key space might produce an unbalanced workload distribution among the nodes in the DHT, while a small key space reduces overall system efficiency. Thus, it is desired and complicated to arrange a moderated key space for the DHT ring where the size of identifier space is optimum. DIS builds a moderated DHT ring considering the VO mode in terms of stability of resources. The stable VOs in the system directly participate in organizing resources in the DHT ring. On the other hand, for the purpose of keeping the size of identifier space moderated, only stable VOs can join the ring through a new DHT node and the unstable VO join DIS by becoming a sub-domain of the other VOs. By using the above strategy and ontology-based query techniques, DIS has extended the DHT capabilities to support semantic querying.

Another drawback of DHTs is the lack of support for dynamic attributes. For example, the dynamic changing resource information of the idle CPU cycles is not possible to be hashed in a DHT overlay. Only the information about the static attributes can be stored in DHTs. Furthermore, DHTs do not support heterogeneity of resources in terms of the number and types of resource capabilities and functionalities, for example, processing resources provide different types of attributes than communication and memory resources, and even the types of attributes for Graphics Processing Unit (GPU) resources might be different from CPU resources. In fact, nodes types homogeneity is an implicit assumption in most of DHTs such as Chord, CAN, Pastry and Tapestry where they expect all the nodes have the same behavior and capabilities. It results in limiting the efficiency and applicability of resource discovery in LDCE saturated by heterogeneous resources.

Topology mismatching or the lack of support for proximity-aware querying is another common problem in DHTs. One of the most important desired features of resource discovery, especially in large-scale computing environments, is the ability of the querying system to discover resources in close vicinity. Pure P2P based systems, like DHT-enabled approaches, generally provide limited (e.g., Pastry) or zero (e.g., Chord) support for such proximity-aware querying. The neighboring nodes in DHT overlay are not necessarily close to each other in the physical topology. The DHT overlay structure destroys data locality, which increases the discovery overhead especially to process a particular type of queries such as resource graph query and range query. Clustering solutions (e.g., super peers) have been proposed to resolve the aforementioned DHT shortcoming (see Section 3.4).

Unstructured Systems: In unstructured discovery, the distribution of the resource information among the nodes is not followed by a predefined or controlled mechanism. An unstructured system can support partial matching and also complex querying. The use of a loose architecture makes the system resistant to node failure and Denial of Service (DoS) attacks. Furthermore, the convenient system adaptation to the frequent node joins and disjoins provides dynamicity for querying in such environment. However, unstructured discovery has a low rate of resource discovery in comparison to structured systems. Nodes in these systems are free to behave as they want and they carry a part of the network functionality. Thus, their failure or misbehavior can be costly. Besides, an unstructured approach has numerous challenges especially in terms of fault tolerance under churn, load balancing, and flash crowds. The unstructured discovery in P2P possess the following distinct properties:

1. Decentralization: implies a system architecture without any centralized server or control point. It avoids single or multi points of failures in P2P environment.
2. Self-organization: contains the capability of the system to be dynamically reconfigurable, which facilitates forming a dynamic network overlay while the system keeps changing over the time.
3. Autonomy: the peers in the system are independent entities, which have autonomy to make a decision based on their preferences and priorities, to establish or cut a connection link to other peers in the overlay network.
4. Anonymity: the peers in the system do not have a global knowledge about the whole system. In other words, each peer only knows about itself, and no one knows about the others (even neighbors).
5. Unstructured: there is no predefined structure or fixed points to store and access the resource information.

Accordingly, depending on the search methods (see Section 4.3), unstructured discovery solutions (e.g., Gnutella [89], JXTA [90], Freenet [89], Morpheus [91, 92] and Routing Indices (RI) [93]) can be categorized in two groups: deterministic and non-deterministic discovery.

Gnutella [94, 95] is a purely decentralized resource discovery based on a pervasive exchange of messages (aggressive flooding) which has initially been designed for file sharing. However, it has been extended to cover for the domain of resource sharing in the computing environments. Gnutella consists of a set of Gnutella Nodes (GN) which are called servants. They play both roles of client and server entities. Each servant is responsible for processing the received queries, check for matches against their local set of resource information and respond with related results. The search mechanism relies on broadcasting and back-propagation communication. It starts by sending a “Query” message contains a criteria string and a randomly generated message identifier, flagged with NHP (number of passed hops) and Time To Live (TTL) fields from a requester to all of the servants in its vicinity. Upon receiving a “Query” message in the target node, it decreases the TTL value in the header descriptor. If the query conditions are not satisfied with the current resources and the TTL after decrement still is not zero, the servant broadcast the query message along the (open TCP) connection links to its neighboring servants, except the arrival link of the query. Additionally, each GN uses a caching mechanism to maintain the track of the recent queries for the purpose of back-propagation of the result messages (or “Hit” messages) to the original requester, and also avoid to duplicate forwarding of the same queries.

GNs must always keep information about their neighbors updated. For this purpose, each node periodically broadcasts “Ping” messages to its neighbors and the receivers answer with “Pong” messages to confirm their presence. GN also supports “Push” message which is a request that can be sent to the resource providers in order to ask them to contact the resource requester. Gnutella provides efficient properties especially in dealing with dynamicity and heterogeneity of resources. However, it has many drawbacks: the search is non-deterministic, the TTL technique implies a better scalability but reduces the number of explored hosts for a query. Especially for the rare resources, the limited query radius leads to the small number of hits (successful discovery). Increasing the TTL on the spot a query is not resolved is a technique to solve this problem, but it results in a logarithmically increase of the network traffic, which could not be a scalable approach in dense communication systems. The other shortcoming is that Gnutella does not support exact matching. The search is based on keyword querying, which is not accurate. The protocol is also not efficient concerning resource consumption and due to its flooding nature (broadcasting) as it invokes too many nodes to handle a query. There are several works in the literature [96, 97] that proposed alternative methods and techniques to improve the Gnutella drawbacks.

Routing Indices (RI) [93] uses distributed indices in unstructured P2P networks. The advantage of this mechanism relies on the fact that queries are disseminated and forwarded only among the places of the network where resources existed, thus avoiding to flood query requests to the nodes which are not useful. The main drawback of this solution is that this indexing system comes from the presence of cycles in the network graph. A recent work [98] of this type extends RI and proposes a technique to perform resource discovery in Grids based on

P2P with the capability to perform multi-attribute queries and range queries for numerical attributes. It uses an information summarization technique presented in [99] and creates different types of summaries and accordingly presents a metric (called goodness function) needed by RIs to guide the query process. It still suffers from RI drawbacks as well as the lack of support for complex querying. A similar proposal [100] presents a task/job-level resource discovery with limited flexibility of querying to handle multi-core machines in Desktop Grids. This technique handles resource availability based on a few set of numerical parameters, such as CPU speed, number of cores, and memory availability. We must note that most of the proposed unstructured discovery solutions in the literature are initially designed for file sharing applications which are out of the scope of our interest in this paper.

Hybrid solutions try to combine and strengthen the benefits offered by the traditional structured and unstructured resource discovery systems. MatchTree [48] and Tripod [49] are examples of hybrid solutions (see Table 2).

Table 3 provides a comparison of major types of resource discovery for both Grid and P2P environments in most aspects. We further discuss the details of the performance factors used for this comparison in Section 5.

3.2.3. P2P-Grid Systems

As we discussed in Section 3.2.2, P2P systems contain autonomous entities that can accomplish actions automatically and without any manual control. Furthermore, they can deal with scalability issues while providing a significant efficiency concerning the dynamicity of resources. These features make P2P strategies as powerful solutions for implementing decentralized discovery in Grid environments.

A P2P Grid consists of a group of Grid peers which share their resources in a purely decentralized manner. A Grid peer contains a set of local nodes where each node (a computing machine) can manage several Grid resources. All the nodes potentially can play the role of server or client. Since the importance of all the nodes is equal, the system does not suffer from a single point failure, and it could be more scalable in comparison to other counterparts. The general search mechanism for a query is to explore the local Grid peer in advance. The unsuccessful query, in the next step, will be forwarded to the closest remote Grid peer in the vicinity using various strategies and techniques (such as Random Walk or anycasting).

Importing and leveraging the alternative concepts like P2P can improve the performance of the resource discovery in Grid [101, 102]. The P2P-based approaches offer a significant advantage over their hierarchical (Grid) counterparts by way of resistance to failure and traffic congestion. In particular, structured P2P systems based on DHTs are very popular for file-sharing applications, but not for sharing resource information. Moreover, typical structured P2P-based systems are very sensitive to churning leading to resource unavailability [103]. These systems achieve good performances and scalability characteristics, but they are limited to only support exact matching. Moreover, their hashing functionalities perform well with static attributes, but they need to be enhanced for handling dynamic

Table 3: Summary of comparison of the major types of resource discovery for different performance factors (PFs: Performance Factors, RB: Reliability, FT: Fault-Tolerance, LB: Load-Balance, AN: Autonomy, CQ: Complex Querying).

PF	Centralized-Grid	Hierarchical-Grids	Structured-P2P	Unstructured-P2P
Scalability	Not scalable especially for the large-scale Grid because of the single point of failure bottlenecks	Provides better scalability than centralized system, but still suffers from overloaded hot spots; the roots of hierarchy becomes easily single point of failure	Limitations due to the significant amount of overhead network traffic	Generally scalable
Efficiency	Provides efficiency in terms of fast discovery and precise discovery results	Similar to centralized	Fast lookup speed. Guarantees to perform a query in a bounded number of hops. It assumes that the availability of the resources in the network is guaranteed. Thus, it doesn't provide a good efficiency to work in a dynamically changing environment. The dynamic changes of resource information must be propagated over the network which reduces system efficiency by creating a considerable amount of network overhead.	Low lookup speed. Costly membership management. Slow propagation of information in the case of dynamic values. Does not assume any guarantees about the peers or resource availability. Therefore it does not put any constraints on the topological placement of resource information.
RB	Highly reliable in terms of result accuracy. Not reliable in terms of single point of failures	Reliable in terms of result accuracy. Not reliable in terms of single point of failures.	Reliable in terms of exact matching. Not reliable in terms of interval search	There is no guarantee to perform successful querying. Reliable in terms of dynamically changing environments
Dynamism	Supported in terms of dynamic attribute. Not supported in terms of indexing mechanisms and periodical updating of the resource information	Supported in terms of dynamic attributes. Better support for indexing mechanisms and periodical updating.	Supported concerning dynamic topology changes. Not supported concerning dynamic changes of attribute values where the system has issues to store rapidly changing resource information. The overlay networks are appropriate to maintain static resource information while for dynamic information the overlay must be reorganized which is costly.	Highly supported
FT	Not supported	Not supported	Supported	Supported
LB	Not supported	Not supported	Supported	Supported
AN	Not supported	Not supported	Supported	Highly supported
CQ	Provides all kind of flexible querying due to the advantage of using central data structure/database	Limited support	Do not support or have a limited support for interval search (such as partial querying and range querying), semantic search, multidimensional querying and resource graph discovery	Better support for complex querying in comparison to structured P2P and less flexibility than centralized systems

objects appropriately. SWORD [104] is a P2P-based approach which supports multi-dimensional resource attributes and range-based querying, and improves the existing systems by resorting to multiple DHTs. ERGOT [86], DIS [88], RI [93], [98], and [100] are other examples of P2P-based Grid discovery which have already been discussed in Section 3.2.2. These Grid approaches, in fact, have tried to improve original P2P mechanisms in order to resolve some inherent shortcomings of DHTs and P2P strategies, as discussed in the previous section.

We can conclude that the combination of P2P and Grid would be desirable, particularly to build scalable and fault tolerant resource discovery approaches for large-scale distributed systems (e.g., [17, 105–109]). Further examples can be found in [110] which is a review of the most promising Grid systems that employ P2P strategies to facilitate resource discovery.

3.2.4. Cluster, HTC and HPC

Computing systems such as Cluster, High Throughput Computing (HTC), High Performance Computing (HPC), and Cloud are generally based on a centralized/hierarchical architecture, leading to the use of centralized resource discovery to respond to user requests. For example, when a request arrives at a HPC cluster-head (or a Cloud service provider in the front-end), the resources required can be discovered (allocated or provisioned) by searching in a specified pool of resources (or in a back-end datacenter). The number and the type of resources are known

beforehand (this may not be necessarily true in Clouds). For such environments, resource discovery based on a centralized architecture could become an easy task. And, in fact, instead of discovery problem, the resource scheduling issues are dominant. In this and the next section, we aim to discuss resource discovery for HPC and Cloud. However, due to the nature of these types of environments, we may also discuss relevant topics such as resource scheduling.

A Computing Cluster can be defined as the composition of several computing nodes (workstations), multiple storage devices and interconnections, which together appear to the users as a single system that is highly available and reliable to run user tasks. HPC, HTC and Many Task Computing (MTC) are some types of computing clusters.

HTC environments provide a significant amount of processing capabilities to run user jobs (i.e., independent loosely-coupled tasks), over long periods of time, through the dynamic exploitation of all the existing available computing resources in the system. In fact, HTC tasks are sequential and independent. They can be individually scheduled on a broad range of heterogeneous computing resources, geographically distributed across multiple administrative domains. Various Grid Computing techniques can be used to build HTC systems [111]. HPC creates supercomputers to run tightly coupled parallel tasks on large amounts of computing capacity over short periods of time. HPC focuses on the fast execution of tasks (generally dependent tasks). There-

fore the interconnects of HPC clusters must provide low latency for communication between processes running on different computing resources. MTC is the bridge between HTC and HPC with an emphasis on employing many computing resources over short periods of time, for executing many computational tasks that could be either dependent or independent.

By using traditional cluster management systems, it might be possible that each cluster nodes knows about the static attributes of the resources in other nodes without relying on a complete approach for resource discovery. For example, ignoring the dynamicity issues, a centralized cluster can statically be configured to maintain all the information about the cluster resources on a central server and the clients just simply submit the queries to the server. In some cases, the static information is not necessary for the task allocation in the cluster. In other situations, this static information might not be enough since the demand might be raised for the dynamic information about resources. The dynamic information can be used to determine whether the required resources are currently free or are occupied. However, state information (i.e., the information about the dynamic resource attributes) such as processor availability, memory usage or available bandwidth, which are rapidly and frequently changing, may not be accessible without leveraging an efficient resource discovery mechanism.

Resources may have multiple single attributes which can be either dynamic or static. Thus, depending on the computing environment and the purpose of resource discovery, different discovery strategies can be implemented, ranging from single-attribute to multi-dimensional discovery systems. CPU availability (CPU load or CPU utilization) and memory utilization (memory usage) are two important dynamic attributes which have the key roles to structure any computational resource discovery mechanisms. In fact, task allocation is not feasible if the qualified discovered resources would be unavailable in terms of CPU and memory load. In general, the resource discovery problem in computing clusters aims to find the available computing resources in the system for job scheduling and task allocations. CPU discovery and memory discovery are the examples of single-attribute discovery (i.e., state discovery) where CPU utilization and memory usage respectively is the only resource information of interest.

In cluster computing, we can categorize resource discovery approaches (in the case of state information) in three groups: Active, Passive and Predictive. In the Active approach (e.g., centralized batch systems and decentralized state discovery algorithms), the requesters use an intrusive method (within a client-server like communication model) to know about the state of the other resources. Additionally, the Active approach does not support dynamic load balancing. In the Predictive approach, the requesters use the state of the resources in the previous cycle (which has already been obtained using an Active method) to predict the state of the resources in the upcoming cycle and thus reduces the overhead of the Active method. The Active approach generates a substantial network overhead due to the periodical resource state updates with direct communication between the resource manager and the compute nodes. The Passive approach attempts to get the information about the state of the resources by analyzing the behavior of the existing network traffic in the

cluster. Since the Passive approach is not an intrusive mechanism, it doesn't generate extra overhead, which results in a decrease of the communication complexity (the number of messages required to solve a discovery problem), but provides less accurate information.

In cluster computing, another alternative for Active discovery is using decentralized algorithms such as ALG-Flooding [112], Swamping [112], Random Pointer Jump [112] and Name-Dropper [112]. These approaches can gather the state information of the resources on different nodes around the network. The state information denotes that if a particular resource is available (concerning CPU or memory) to allocate a specific task. We discuss these algorithms in more details in Section 4.3.

The Batch System (or Asymmetric Computational Clusters) [113–116] is a conventional type of cluster management systems which has been used to compose most of the current large-scale computational clusters. It contains three types of essential components: Resource Manager (providing resource capabilities and status), Queue/Job Manager (including creation, queuing, controlling and monitoring of the user's jobs) and Scheduler (containing resource mapping and allocation) (see Figure 11).

The queue manager lets users submit their jobs to the queues, and provides possibilities for users to monitor and control the state of their jobs, which can be running on the computing nodes or waiting in the queues. For each task, the scheduler selects the target computing nodes and then assigns the jobs to the resources on the computing nodes according to the scheduling policy. In the next step, the resource manager monitors the state of the resources on the computing nodes as well as the state of the job executions in the resources. The computing nodes periodically update the resource manager regarding the state (e.g., available, occupied) of their resources or the resource manager can be responsible for getting this information through periodical querying of the computing nodes. Examples of batch systems include Condor [54–56], Berkeley Open Infrastructure for Network Computing (BOINC) [117], Globus [118], Fast and Light-Weight Task Execution Framework (FALKON) [119], Oracle Grid Engine (previously known as Sun Grid Engine or SGE), Tera-scale Open-source Resource and Queue Management (TORQUE) [120], Dodo [121, 122], Slurm [123], and PBS [124].

Condor [54–56] is a centralized batch system that is designed specially for HTC clusters. Like other batch systems, it has three main components: User Agents, Resource Agents (or Resource Owner Agents) and Matchmakers (see Figure 12). User requests are represented within User Agents, so the User Agents submit their resource requests to the Matchmakers which includes all the constraints that define the characteristics of the desired resources. For example, a User Agent (requester) may need only to find the resources running a particular operating system. On the other hand, resources are represented within Resource Agents, which contain the resource owners policies. For example, a resource owner may only be willing to serve the requests made by a particular group of users. The Resource Agents advertise their resources through resource offer messages to the Matchmaker where it attempts to find the matches between resource requests

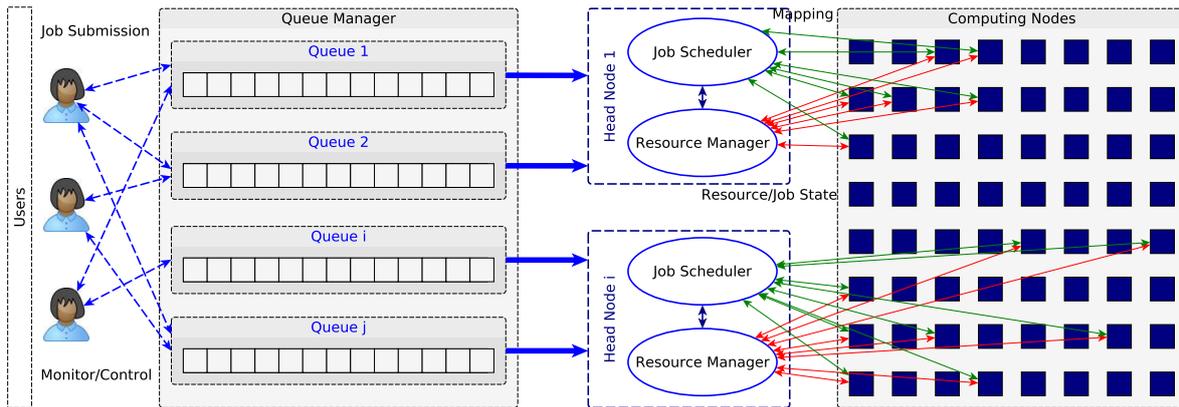


Figure 11: Computing strategy in Batch Systems.

and resource offers in a way that all the constraints of the three parties (User Agent, Resource Agent, and Matchmaker) are satisfied.

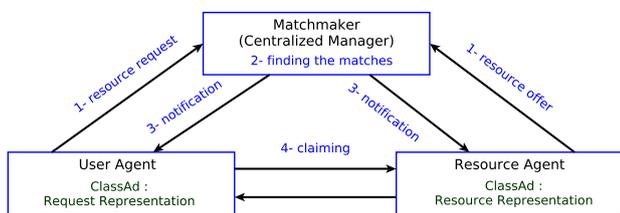


Figure 12: Condor matchmaking.

The Matchmaker [54] also implements a collection of system-wide policies including logic to map the resource requests to the resource information through imposing its constraints in the matchmaking process. For example, the Matchmaker generates the priority ranking for the requests. Thus the requests with higher priorities have greater opportunities to find the matches. When a match is found, the Matchmaker notifies both User and Resource Agents, which consequently will execute a claiming protocol to start the task allocation process. The Matchmaker is also able to measure how well a resource can satisfy a request by running a preference function. The resource information is expressed in ClassAd format, which is used by the Matchmaker to facilitate the matchmaking and resource preference measurement. Additionally, Condor provides features to support resource scavenging: a capability to locate idle resources using a daemon providing summarized monitoring information such as average load, total idle time, as well as some other elements of basic information.

BOINC [117] is a centralized loosely coupled HTC platform (i.e., desktop Grid like Entropia [125]) that provides resource sharing facilities for large number of volunteers to create, operate and monitor the public-resource computing projects such as *@home (e.g., SETI@home [126], Predictor@home, Folding@home [127] and Climate@home). BOINC encourages the creation of many projects while it encourages volunteers (the computer owners around the world) to participate in one or more projects. Resource owners can enforce their policies to specify how their resources are allocated to each project. Each BOINC

project is identified by a single master URL which serves as a directory of scheduling servers, while resource owners can register their resources and participate in the projects. BOINC, similar to other batch systems like Condor, relies on knowing state information of all computational resources, which results in a large amount of information that limits system scalability. In fact, BOINC suffers from a scalability problem since it uses a global scheduling mechanism.

TORQUE [120] is a Workload and Resource Management System (WRMS), which consists of two components: one for job execution and resource monitoring, and other (called Moab) for queue management and scheduling. These two components together provide all the necessary functionalities and meet the requirements of a batch system. TORQUE uses a one-to-one mapping scheme to allocate jobs to each compute node available. This mechanism is not efficient because each compute node can perhaps execute more jobs. TORQUE is also not able to dynamically balance jobs among resources. FALKON [119] is derived from TORQUE aiming to enhance its efficiency, enabling to assign multiple tasks to a single compute node. However, like TORQUE, it doesn't support dynamic load balancing. FALKON integrates multi-level scheduling and streamlined dispatchers to enhance the system performance by decreasing the task execution time. It attempts to achieve higher scale through renunciation of some functionalities such as priorities and multiple queues (which already have been provided by Condor). However, since the FALKON dispatcher is centralized, its scalability is limited.

Dodo [121, 122] is a batch system for harvesting idle resources (in terms of memory) in off-the-shelf clusters of workstations which include the components such as, central manager, scheduler, resource monitor and the idle memory daemons. In this system, resources periodically inform the central manager regarding their state (free or busy). The system only schedules the jobs to the resources on the idle compute nodes. Therefore it can not provide support for dynamic load balancing.

Simple Linux Utility for Resource Management (SLURM) [123] is an open source batch system which is widely used by many supercomputers and computer clusters around the world. It provides a simple, fault tolerant, and highly scalable resource management system for large-scale Linux clusters including

thousands of nodes. It is also flexible to be ported to clusters with different size and architecture. However, SLURM does not provide a comprehensive cluster administration/monitoring capability. SLURM is not a sophisticated batch system since it only provides high-performance parallel job management while leaves most of scheduling decisions to an external entity. In fact, it does not even perform functions such as general purpose event logging or historical state recording for its compute nodes. The SLURM's default scheduler implements First-In First-Out (FIFO).

Portable Batch System (PBS) [124] is a commercial batch system, providing a flexible batch queuing and workload management solution for HPC systems and Linux clusters. It allows sophisticated scheduling by extracting various scheduling policies, customized by site administrators. PBS spawns daemons on each machine and provides additional controls for initiating/scheduling processing jobs on host machines or routing jobs between different hosts. While PBS provides a portable and flexible resource management approach, it has significant drawbacks. PBS is single threaded, resulting in a very poor performance on large scale clusters. PBS also suffers from a weak mechanism for starting and cleaning up parallel jobs. LSF [128], Load Leveler [129], and CCS [130] are further examples of schedulers and resource managers for HPC systems.

3.2.5. Cloud Computing Environments

Cloud computing [1, 131, 132] provides infrastructure (Infrastructure as a Service or IaaS), platform (Platform as a Service or PaaS) and software (Software as a Service or SaaS) as services to the end users. It helps the users to create and manage their customized hardware and software, while it reduces the cost and the complexity of maintaining the required hardware/software for different user's applications. Grid and Cloud computing attempt to highly utilize and manage distributed resources to attend large (computational) jobs efficiently. But, Grids and Clouds are fundamentally different in several aspects. Clouds have built-in administrative boundaries [1, 133] which affects its capabilities to be inter-operable, while Grid do not reflect such limitation in its operation[134]. Grids nodes have autonomy to self-manage their characteristics and behavior.

Cloud computing uses negotiated Service Level Agreements (SLA) to dynamically scale the user instances in the hardware infrastructure, software platform, and software application according to the potential resources in the Cloud. However, the capability of a single Cloud is limited, and it might happen that a single Cloud is not able to uniformly maintain the quality of services for all users requests. One of the ideas to overcome this problem is the convergence of the Cloud with Grid along the line of the inter-Clouds federation. It must be taken into account that one of the most important integral element in this convergence procedure is the problem of resource sharing, which contains the issues related to information dissemination, resource matching, and discovery.

The resource scheduler in the (computing) Cloud receives the user request, which includes a set of required hardware to build a customized computing system. Afterward, the scheduler starts to look up a matched set of physical or virtual resources.

If the demanded resources are not available in the system, the Cloud service provider (system manager) has to create virtual resources (resource/service provisioning) which are precisely matched to the request. So despite the Grids super-schedulers (Grids controllers), Clouds schedulers are responsible for both resource discovery and resource provisioning. In Grids, each Grid site may have a super-scheduler (or global/meta scheduler) which is in charge of allocating jobs received from clients to the global/local resources found by resource discovery.

Clouds are based on a Service Oriented Architecture (SOA) where each service is accessible through a broker. The clients submit their requests, containing the required QoS levels for the desired services to the brokers. Consequently, the brokers proceed to find and allocate the best matching service provider for the requested services with a certain level of QoS. There are a significant number of commercial brokers such as Cloudswitch [135], Deltacloud [136–138], Eucalyptus [139] and Elasta [140, 141] which already are used in different Cloud systems. However, there is still no standardized method to map Cloud services to the clients' requests, based on QoS level.

Considering the administrative boundaries of the Clouds and the fact that Clouds allow resources that can be elastically divided and reassembled to meet the end users requirements, the concept of resource discovery in Cloud differs from Grid. Here it has to support scheduling and resource management functions such as resource provisioning, resource brokering, resource mapping, resource allocation, resource modeling, and resource adaptation. Resource discovery, hence, is especially emphasized when we move toward inter-Clouds approaches or Self Organized Clouds (SOCs). Each single Cloud provider or individual host are required to autonomously locate and discover a set of qualified volunteer computing resources in the network for its job's execution via different types of querying strategies. Since *i*) we want to focus on resource discovery, and we are not going to discuss in depth other relevant resource management aspects (like resource provisioning) in Cloud computing (there is a comprehensive survey to address the resource management issues for Clouds in [142]); *ii*) due to the point that most of resource discovery solutions for Grid and P2P are also applicable to be used in Clouds; we focus in this paper on the Cloud architectures and discovery solutions which have specially designed for multi-Clouds environments such as Federated Clouds, Hybrid Clouds, and Self-organized Clouds.

The increasing number of Cloud services along with the massive amount of global users and inherently limited scalability of the current single provider Clouds became the reason for the development of multi/many providers Clouds. In such an environment, called a federated Cloud, resource discovery becomes a critical issue to address the federated placements, considering the point that all Clouds are not equal in terms of capabilities, performance, QoS, availability warranties, and cost. The federated placements refer to the process of clarification of the most appropriate cluster to use for a particular application workload. Along with this line, there are a number of resource-centric IaaS providers such as Eucalyptus, OpenNebula [143] and Nimbus [143] that provide the clusters of virtual machine hosts as IaaS resources. They offer the list of their resources types con-

taining the prices and the detail of capabilities for each particular resource type.

RESERVOIR [144, 145] introduced the notion of federated Cloud. The Federated Cloud contains several single Cloud providers joined by a mutual corporation agreement which makes inter-Cloud computing and cross-Cloud migration feasible in a cost-efficient manner. Providers that have excess capacity can share their additional (available) infrastructure resources with the federation members that need those resources to overcome the problems such as resource limitation and over-provisioning. RESERVOIR is an open federated Cloud computing model, architecture, and functionality which aims to deal with scalability, complexity, and interoperability in multiple Clouds environments.

In the RESERVOIR model, two or more Cloud providers join the system to create a federation Clouds. This way, various Cloud services/resources can dynamically be managed and controlled together. Logical services/resources are represented and encapsulated in a Virtual Execution Environment (VEE). VEEs are hosted on top of physical resources and virtual application networks which are represented as Virtual Execution Environment Hosts (VEEHs) managed by a Virtual Execution Environment Management System (VEEM). The VEEM is responsible for performing the management, deployment, and migration of VEEs on top of VEEHs through the utilization of OpenNebula. Moreover, the service manager component is responsible for instantiating services and manage the SLA. RESERVOIR provides facilities to describe and specify the application configuration, which results in defining a service by a set of information such as the specification of the virtual machine, application configurations, and deployment settings. RESERVOIR attempts to maximize system flexibility to address the different requirements and policies from various infrastructure resource providers, by supporting dynamically-pluggable policies to measure and calculate the placement. Different policies define different utility functions which have to be optimized. For example, a load balancing policy aims to distribute the Virtual Machines (VMs) equally between physical resources, while an energy saving policy tries to minimize the number of physical resources for VM allocations; thus, to reduce energy consumption, the unused machines can be turned off. The RESERVOIR Cloud model enable the individual Cloud providers to have a focused view of resources, while fully preserving the individual autonomy of the providers in making technological, policy and management decisions, by leveraging and combining the advantages of virtualization and embed autonomous control. This helps the resource/service providers to completely describe and define their resources that help to implement efficient and accurate resource discovery approaches along the line of Cloud computing requirements.

Another example of resource discovery, for multi/many providers Clouds, is presented in Wright et al., [146]. It deploys a two-phase constraints-based resource discovery solution which uses a software abstraction layer to discover the most suitable infrastructure resources in a multi-provider Cloud environment for a given application request. In the first step, the discovery system identifies a set of potential infrastructure resource candidates

who can satisfy the application requirements concerning quality of service. And in the second phase, an appropriate heuristic method depending on the application preferences (for example, some application may prefer to use a cost-based heuristic while others can prefer the performance-based heuristics) is used to select the best matching candidate among the initial set of discovered resources. The proposed model shows a dynamic flexibility to choose the resources based on the application requirements and preferences.

3.3. *Ontology and Resource Description*

In a computing environment, there are several different entities (e.g., resource providers, resource requesters and directory agents) that are seeking a common purpose of resource sharing. For this purpose, at least they should have a global understanding of resources, which requires a shared definition of resources in terms of ontology (i.e., description/definition of resources). A resource description model facilitates collaboration among system entities and reaching an agreement to perform resource sharing through discovering, mapping, allocation, and invocation of resources.

Resource discovery and mapping can be simply defined as the process of finding computing resources and efficiently mapping computing applications (or application segments) to the discovered resources. For each application query, the discovered (selected) resources are the best matches according to the application requirements while the resources in the system must be utilized efficiently. For doing this, it is required to abstract the complexities of the system components such as computing applications (application description) and hardware computing resources (resource description) through a consistent methodology and language which describe their characteristics, requirements, and capabilities. In fact, the resource capabilities can be described and derived from hardware descriptions. According to the application description, the discovery system must be able to exploit the application resource requirements to generate a proper query. In the next step, the query explores the network to find the best matching resources through the evaluation of the resource description of each resource.

Distributed resource discovery in a large-scale system requires a scalable and fully expressive resource description, which contains a required level of information details in terms of computational (e.g., processor description) and communicational (e.g., topology description) properties and behaviors. Most of the resource discovery behaviors and operational characteristics are under the influence of the way we abstract, organize and distribute the individual resource information. For example, in a typical P2P system, for the purpose of resource discovery, each one of the peers has to get information from other peers and disseminates the information to others through neighbor peers. In this case, it is important to consider the point where the resource information has to be distributed and get balanced between peers. It helps to improve the scalability of the system when the number of peers grows increasingly. Moreover, resource description has direct impacts on the indicators such as the discovery latency (response time), the accuracy of results (rate of false discovery), the discovery traffic/overload, etc. We

categorize the approaches for resource description in two groups: attribute-based and semantic-based schemes.

The attribute based schemes define the resource characterizations through a set of (attribute, value) pairs. Depending on the level of information details and the different storage, retrieval and distribution mechanisms these approaches can provide a scalable distribution of resource information. However, the attribute-based description models are facing some challenging issues such as providing appropriate support for dynamic and collective attributes [147]. Dynamic attributes are frequently changing, which results that the description models mentioned above become unappropriated to store their values due to expensive updating and maintenance cost. On the other hand, all resource properties are not independent. Rather, they are related to each other in some aspects (depending on the level of abstraction). Therefore, it would be more complicated to exploit all resource properties and capabilities individually in the form of (attribute, value), without leading to information redundancy and inaccurate resource description.

XML, Web Services Description Language (WSDL) [148], GSDL [149], OASIS Universal Description, Discovery and Integration (UDDI) [150–152] and [153] are some examples of attribute-based resource descriptions which have been used in many resource discovery solutions specially in web-based resource discovery protocols. WSDL is a set of instructions to describe the behavior, characteristics, and formats of services, particularly for web service providers. UDDI uses a XML-based repository to provide policies and standards for service discovery which facilitate the process of resource advertisement and service publication. Tutschku et al., a recent work [153], is proposing a resource description method based on the capabilities of on-board Linux tools for describing resource utilization in cloud networking and NFV infrastructures. It aims to provide a description approach for dynamic attributes such as CPU load and usage. However, the approach is not general (it is based on Linux) and is limited by a very abstract description of resources and also the description is restricted to a very few number of predefined general attributes. In addition to the attributed-based description languages, there are a number of recent attributed-based (resource information) encryption methods in the current literature including [154–159] which are mostly application-oriented. And in fact, they are not flexible and even efficient to be used for different applications.

Semantic-based description models are the alternative approaches which focus on the overall collaborative description of the resources. These strategies are adequate to describe all system resources where all the possible collaborative system and resource properties and behaviors (e.g., the structure of the processor or memory architectures) are precisely described, but we must take into account that these approaches might not be scalable concerning the distribution of the resource information. The works in [160–163] are examples of semantic-based schemes which use functional languages to describe hardware resources. The use of higher-order functions allows the composition of arbitrarily complex structures in a clear and concise way. The strong type system of most functional languages also ensures the soundness of the composition of the different hard-

ware components. Functional resource description models focus on capturing the structure as well as numerical properties of hardware resources. Resource descriptions themselves are functions, capturing the fact that behaviors/capabilities relevant for a resource can change under certain circumstances. Additionally, functions can be used to concisely and clearly capture complex parameterizable collaboratives. For other examples of semantic-based ontologies we can mention Resource Description Framework (RDF) [164–166], Ontology Inference Layer (OIL) [167], DAML+OIL [168–171] and DAML-S [172].

RDF is a type of ontology/knowledge representation approach, which is a primitive language providing a binary relation of classes and properties supporting all kind of range/domain constraints and sub-property/sub-class relationships. It is a powerful and more expressive language to describe resources in sufficient details. OIL is an extension of RDF while DAML+OIL is an ontology representation language derived from the work under DARPA's Agent Markup Language (DAML) [177] (based on the OIL language). In comparison to OIL, it provides a larger interoperability on the semantic level through extending the OIL and RDF basic primitives along the way to provide an ontology-based description language with better expressiveness and capability for inference creation. DAML-S is another work under the DAML program which provides a set of principles, basic concepts to describe and declare services/resources, by implementing the ontology structuring mechanism of DAML. Each DAML-S service/resource is characterized in three types: service/resource model (process model), service/resource profile and service/resource grounding. These describe the service functionalities (i.e., the service composition and the service behaviors on the run time), the service capabilities (i.e., the required information for resource/service discovery), the service access (i.e., the service address or the required information for service/resource invocation) respectively. In fact, resource model and resource grounding provide the required information for the interaction between resource providers and resource requesters, while the resource profile facilitates the process of matchmaking for resource discovery. There are several proposals such as [178, 179] that utilize DAML-based resource description for resource discovery.

The Lexical Bridge [180] is a recent work which proposes a methodology to translate meaningful information in natural language sources into a standardized, structured knowledge representation for semantic normalization, integration, analysis, and reasoning. However, it is a very general work and in fact, doesn't provide a complete resource description language. Rather, it aims to build "lexical bridges" (LBs) for filling the gap between the natural languages and their ontology representations. The authors in [181] have highlighted that a scalable resource description and information exchange (concerning the distribution of resource information between Clouds) is an essential requirement for sharing heterogeneous Cloud resources among federated Clouds. However, the work presented in this paper, a semantic-based resource description model for inter-clouds, does not provide a scalable solution for distributing all details of resource information in the entire system. It focuses on providing a scalable information exchange method between multiple

Table 4: Examples of grouping approaches for resource discovery (OA: Overlay Architecture).

OA	Mechanism	Grouping	RD Examples
Semantic-Aware	Nodes/resources with similar contents are organized in the same group	Content-based grouping, SON [87]	ERGOT [86], and [46]
Proximity-Aware	Physically nearby nodes/resources are organized in the same group	Content-based grouping	CycloidGrid [50], TriPod [49], PIRD [51], PIAS [173], and ASTAS [174]
QoS-Aware	Nodes/resources with similar quality of service are organized in the same group	Content-based grouping	CycloidGrid [50]
Load-Aware	Nodes/resources are organized in a particular overlay (e.g. DHT ring) in order to equally distribute the loads (in terms of resource information or query workloads) among nodes to enhance the overall system performance.	Non-Content based grouping, P2P overlay network (e.g., DHT based P2P)	PIAS [173] and ASTAS [174]
Super-peer	Nodes/resources are organized in a particular overlay (e.g., tree, hierarchy) to enhance the overall system performance.	Non-Content based grouping, Non-DHT based P2P Overlay	The works in [105, 175, 176]
Tree	Load balanced Tree Overlay	Tree	MatchTree [48]

clouds, where each cloud centrally manages its resources.

Referring to the approaches discussed above, any optimum resource description model for resource discovery must be designed in such a way that it takes the concerns of both approaches (attribute-based and semantic-based): capturing individual and collaborative capabilities of resources while still allowing for the scalable distribution of this information [181]. Consequently, a merger of the two above abstraction concepts should be pursued.

3.4. Grouping Approaches (Virtual Clustering)

In Section 3.2.4, we discussed resource discovery and management approaches for real (physical) clusters including HTC and HPC. A physical cluster can be defined as a collection of servers (physical machines) connected by a physical network topology. In this section, we aim to discuss approaches that can be used to create virtual clusters/groups (overlays or virtual topologies) on top of physical clusters/systems for the purpose of resource discovery. While the term "clustering" has commonly been used for similar discussion in the literature, we understand that this might be misleading for our discussion in this section. Thus, for the sake of clarity, in this section, we use terms "virtual clustering" and "grouping" interchangeably instead of "clustering". Nevertheless, here the term "virtual clusters" should not be confused with the concept of virtual clusters which are built with virtual machines.

Virtual clustering is the process of creating virtual groups (overlays) and altering the underlying physical network topology to increase the overall system performance. Nodes and resources are grouped in virtual clusters, which share common specifications, properties, operations or behavior. Overlay construction and grouping enhance the efficiency of the query/information management and resource advertisement for the resource discovery systems. It must be taken into account that the system overlays have impacts on a set of important discovery performance factors such as scalability, efficiency and even reliability and dynamicity. Grouping approaches may provide some inherent features like locality-awareness, which could bring benefits to enhance the discovery mechanism. Furthermore, grouping strategies can be used to overcome some shortcoming of DHT and P2P based designs. Examples of these advantages include the ability to support proximity-awareness and semantic-awareness querying while purely P2P based approaches do not support these features. Besides, grouping mechanisms can enhance the degree of resource-awareness in the network, resulting in fast

and efficient resource discovery. By virtual clustering, the grouping of the nodes in the system can be performed based on some similarity between resource characteristics of nodes, meaning that each group (group-leader/group-user) already have some predictions (knowledge) about the potential resources that might be offered by group members. This facilitates the resource discovery procedure by reducing the search space. The search is only conducted in the groups where they can potentially offer the desired resources for a given query, instead of searching all groups in the system. Grouping also matters for automatically creating the underlying virtual structures required for the process of resource discovery. For example, for developing a hierarchical resource discovery, it may be required to build an underlying virtual hierarchy in a dynamic manner, using a grouping strategy.

Overall, we can classify the grouping methods (see Table 4) for resource discovery in three main groups: quantity-based (non-content based), quality-based (content based) and hybrid grouping.

In quantity-based grouping, overlay construction involves the methods to organize nodes/resources in the groups without considering the content (for example, in terms of attributes, features, properties, behaviors) of nodes/resources. For this purpose, the focus might be on some performance related issues such as load balance, maintenance, self-organization, stability (churn and fault tolerant), while the overlay must be constructed along the way that finally satisfies all the requirements of a resource discovery protocol which aims to run on top of it. Super-peer is a type of quantity based grouping.

Super-peer based discovery systems aimed to provide an optimum solution to achieve a balance between the built-in efficiency of the hierarchical/centralized system, and the load balancing, self-organization, and fault-tolerant features provided by P2P based discovery systems. Kazaa [182] is an example of the super-peer model which designates the more stable and robust nodes as super-peers. The new member must find the closest existing super-node in the network and establish the overlay connection to that node. Kazaa has initially been used for file sharing purposes. However, its concept can be extended to use for general resource sharing, even of computing resources. Upon joining a new node to the overlay, it sends its list of resources to the super-node. The super-node registers the resources information of the individual nodes in an indexed directory. Each requester sends its requests to the super-node. Afterward, the requester searches for the required resources in its local index

directory and, if the desired resources are not locally available, it propagates the query to another supernode in the system. Kazaa is also able to perform the keyword search and provides some flexibility for querying. The works in [105, 175, 176] are other examples of super-peer based discovery systems. These solutions enable efficient discovery of the resources belonging to one super-peer. However, they do not introduce any further improvement for P2P based resource discovery over several super-peers. section

Quality-based grouping approaches organize the groups based on the content similarity of each node/resource in different aspects. According to the different views and definitions of the concept of the content similarity, there are several well-known approaches for content grouping which shapes the discovery mechanisms in the directions. Proximity-ware [183–187], semantic-aware [188–192] and QoS-aware [193–195] resource discovery are the sample applications of content-based grouping. Proximity-aware grouping approaches (like TriPod [49]) organizes the close resources in the same group while semantic-aware grouping methods [46, 196] organize the semantically similar resources in the same group. Depending on the strategy, virtual clustering can preserve both locality and similarity features.

QoS is one of the most important aspects of resource discovery systems in large-scale environments, especially in Grids and Clouds [197]. It must be taken into account that the QoS concept is not only limited to network capabilities like network bandwidth, rather it is extended to all kind of computing and storage capabilities. In fact, when a task with QoS conditions is submitted to a Cloud or Grid system, it would be necessary to negotiate a SLA to guarantee the quality of the requested service according to the QoS conditions. Accordingly, resource reservation is one of the best mechanism benefiting from QoS. As an example of QoS-aware resource discovery, the MDS Globus provides this kind of resource reservation capability [16]. However, this reservation has a poor efficiency to guarantee the bandwidth for the network links which results that the system would not be able to secure the reservation of the processing cycles for the application segments that communicate over those network links.

As we described in Table 4, QoS-aware grouping is a type of content-based grouping where resources with similar quality of service are organized in the same group. Control Groups (Cgroups) [198] is an alternative grouping solution, designed for Linux systems, which can guarantee QoS for resource allocation and scheduling. However, in contrast to our discussion in this section, this approach provides a fundamentally different concept of grouping. Cgroups are user-defined groups of tasks (processes running on a system) which are configured manually by the user. In fact, the grouping is based on the manual configuration of tasks instead of automatic self-organization of resources. Accordingly, hardware resources can be appropriately divided up and allocated to the defined Cgroups of tasks.

Currently, most of the Grid/Cloud systems provide only a partial solution for QoS, due to the point that the majority of the processor operating systems do not provide explicit performance guarantees. The partial QoS solution means that the system

provides the capability to specify QoS conditions at the time of job submission while it does not have support for resource reservation mechanism.

4. Design Aspects

For designing a resource discovery approach, it is required to specify which design choices to make for different parts of the discovery process. A resource discovery system, first, must be initialized through using a bootstrapping mechanism. We must also specify an overall discovery strategy. According to the overall strategy of the discovery, an adequate search method must be applied for directing and processing the queries in the system. In this way, it might be necessary to specify a method for the propagation of queries in the system. Furthermore, it is important to provide a strategy for information delivery by queries (each query may contain certain types of information which are delivered between different entities in the system). A discovery approach also needs to provide a mechanism for query termination. Otherwise, a query may search forever, or multiple computations of a single query might happen in different places of the system, due to the same query arrival through different discovery paths.

4.1. Bootstrapping

Bootstrapping is the process that occurs when a new element joins the discovery systems. The new node may not have information about the other nodes/resources in the system overlay. Thus, in the first step, it is required to discover a node (bootstrapping node) that already belongs to the network overlay, or at least has some initial information about the system. This preliminary information may include the address of the resource provider nodes, the super-peer nodes, the directory agents, the server nodes, the neighboring nodes or even the multicast address of the group. In other words, the new nodes will join an existing system through asking a bootstrapping node for a list of already known nodes to which it uses later to perform a resource query.

There are several mechanisms [199] for bootstrapping. In a small network, the new node can announce its presence through conducting a simple flooding. Since flooding is not efficient in larger networks, multicasting can be used instead of flooding. In centralized structured systems, it would be easy to use a service like Domain Name System (DNS) [200], thus, the new nodes can easily use the DNS to get the server address. In traditional service discovery systems [201] like SLP [202], both service and user agents try to find a directory agent (Active method) for either service announcement or service discovery. The Passive approach lets strategic nodes such as directory agents or bootstrapping neighbor nodes to periodically announce their information in the system through multicast or even broadcast. Generally, in decentralized and in unstructured systems, the new node asks the bootstrapping neighboring nodes for information about the system overlay. The bootstrapping nodes can be either a central registry server or any node that already belongs to the network. The problem is that the central registry server

potentially becomes a single point of failure and, in the lack of bootstrapping node in the neighborhood, a non-bootstrapping node may not provide enough information to determine the positioning of the newcomer nodes in the system overlay. Other possible techniques for bootstrapping in P2P environments can rely on mechanisms to determine the initial neighbors, such as caching (i.e., using the previously stored list of active nodes), and local network broadcasting [192, 203].

4.2. Discovery Strategies

Discovery strategy specifies the overall approach for sharing resources information across the system. A search method for handling queries can be designed with respect to an overall discovery strategy. In general, there are three types of discovery strategies (i.e., modes of discovery operation): Reactive (pull-based), Proactive (push-based) and Hybrid.

In the Reactive mode, the requester creates a query and sends the query to either a resource provider or a resource directory agent. Since the destination of the queries is already known, it is not necessary to flood the system. In the case that the directory agent or resource provider does not exist, the search range can be expanded by increasing the number of hops. Therefore the requester would be able to find the nearest resource providers or directory agent. This can be done by using different propagation mechanisms such as unicast, multicast or broadcast. The requester can simultaneously send the query to one or multiple resource information providers. The Reactive mode avoids flooding by eliminating advertisements. The drawback is that query messages take longer to find resources, due to the blind nature of the search mechanisms.

In the Proactive mode, the information providers (resource providers or directory agents) periodically advertise their resource information to the environment. Therefore every node in the system can update their information. Using this information helps the requesters to resolve most of the queries locally. However, it might be possible that, in a given time frame, the local resource information is not up to date and leads to invalid discovery results, decreasing system accuracy. On the other hand, system maintenance based on periodical updates (i.e., broadcasting the resource information to other nodes) is not cost-efficient and creates a significant amount of network overhead.

The Hybrid mode uses the advantages of both Reactive and Proactive mechanisms by combining and aggregating these mechanisms for all kinds of discovery components such as resource providers, resource requesters and directory agents. As an example of Hybrid mode, we can mention cluster-based resource discovery protocols, which use Proactive mode within the clusters (intra-cluster discovery) and invoke the Reactive mode for searching between clusters (intercluster discovery). When the number of resource requesters is significantly more than the number of resource providers, the Proactive mode provides better performance in terms of latency and overhead while for the environment which has a large number of providers with few requesters the Reactive mode is more efficient [204, 205]. However, in both cases, the Hybrid mode provides better performance [206].

4.3. Search Algorithms

Search techniques are the most challenging part of resource discovery systems. There are various techniques to discover and locate resources in the network, which are differentiated based on their algorithms and the target usage environment [3, 4, 10]. For example, in a small network, with a limited number of resources, no complex search method is required. A node can simply discover other resources by using basic broadcasting or multicasting. Directory-based or centralized systems with a limited number of servers also do not need a complex propagation method for querying. However, in large distributed networks, such as unstructured peer to peer overlays, to support complex or free-form queries, appropriate search techniques have to be applied and integrated with the query propagation methods to increase efficiency and scalability. In this section, we discuss some of the most well-known algorithms which are used for resource discovery in large-scale systems. For this purpose, we classify search methods, in the literature, along with different dimensions such as informed vs. uninformed, synchronous vs. asynchronous, deterministic vs. non-deterministic and bio-inspired vs. non-nature inspired (see Figure 13).

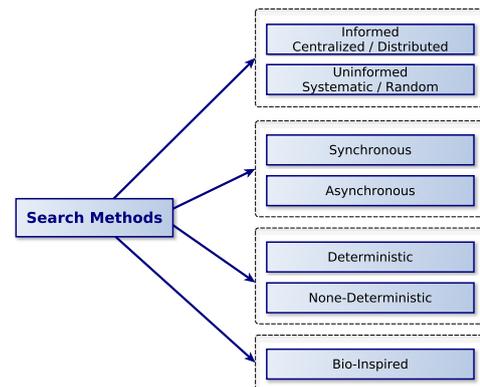


Figure 13: Search algorithms for resource discovery.

4.3.1. Informed vs Uninformed

According to the system overlay and the distribution of the resource information, we can classify the search methods in two groups which are informed and uninformed (blind) search methods [207]. In the uninformed search approach (blind search), the sender node knows nothing about other nodes and resources in the network, while in the informed search approach, each requester or intermediate node at least has some prediction about the location of the resources requested. Uninformed methods can be either based on systematic or random algorithms, wherein systematic approaches, almost the whole or part of the search tree or graph is explored. In random based algorithms, there is an option to reduce the size of exploring space by randomly choosing the next node or invoking some probabilistic technique to disseminate the query.

The underlying concept of most of the blind (uninformed) approaches is based on the reduction of communication complexity, resulted from resource information replications. The complexity can be reduced by limiting the spread of the queries

through mechanisms such as setting small or dynamic time-to-live values for query dissemination, forwarding the queries to only a random chosen subset of nodes or by implementing different strategies for resource information replication like path replication and uniform distribution across the network. On the other hand, limiting the spread of the queries may increase the time complexity of the search method, which leads to slow resource discovery. Query replication may improve the time complexity. Following, we discuss some of these approaches in more detail.

ALG-Flooding [208–210], Breadth First Search (BFS) [211–213] and Depth First Search (DFS) [214–216] are the most well-known systematic search methods. Moreover, there are several other systematic and random search methods such as Depth Limited Search [217], Iterative Deepening [218], Uniform Cost Search [219], Random Walk [220–222] and Gossip based search methods (e.g., Newcast Gossiping [223], works in [224, 225]). Referring to the aforementioned solutions, there are various resource discovery protocols that integrate them with their own query propagation methods such as Probabilistic Flooding [226], Forwarding Protocols (e.g., Selective Forwarding, Intelligent Forwarding and Probabilistic Forwarding) [227, 228], Gossip-based Probabilistic Forwarding [229–232], etc.

In BFS, the search algorithm is deterministic where the algorithm ensures that if the resource is already located in the system, it will be found as the result of resource discovery. The search process is initiated by the source requester, which propagates the query to all of its neighbors. Consequently, the receiving nodes forward the query to all of their neighbors, except the one where the original query came from. Additionally, intermediate nodes avoid further flooding of a query, which has already arrived and processed earlier. The query terminates either if the query is resolved or if there is no edge that query has not passed. The weakness of this method is its huge discovery overhead to find the required resources (due to the costly nature of the flooding) (see Figure 14).

Modified Breadth First Search (MBFS) [233] is a variation of BFS in which the nodes only select some of their neighbors for query forwarding (see Figure 14). In comparison to BFS and ALG-Flooding, it reduces the number of transmitted messages to resolve a query, but it still suffers from large traffic overhead, which comes from its flooding nature.

In the Standard Random Walk solution [234], the query message (Walker) is forwarded to a randomly chosen neighbor at each step until the required resource is found. It leads to a significant reduction in the message overhead, but the solution is slow and requires a large number of hops to resolve the query (see Figure 15).

The K-Walkers Random Walk solution [234] reduces the discovery latency (related to the number of hops) in the standard random walk by increasing the number of walkers. The requesting node sends out the query messages (walkers) to randomly selected k neighbors. Consequently, in each round, the receivers (intermediate nodes) forward the query messages to a randomly chosen neighbor. The walkers proceed to walk to the next nodes until the required resource is found or the query is expired, which happens after a certain number of hops (see Figure 15).

Query termination can be based on either the TTL method or by an explicit verification process. The advantage of this algorithm is its significant overhead reduction in comparison to other pure blind search methods, while it is faster than standard random walk. A common problem affecting most of the blind search algorithms (such as the random walk search) is message duplication, which happens in the case that a node sends duplicate query messages to other nodes regarding the same query. This creates a significant unnecessary traffic over the network. Moreover, the random walk approach is non-deterministic, which means its success is not always ensured, and the rate of successful discovery would be small for the rare resources.

The informed search methods can be classified into two groups: mechanisms that set specific nodes to act as index nodes (index servers), and mechanisms where indices are distributed among all the nodes (i.e., each node can be an index node). The approach employing specific index nodes has been used in most of the resource discovery solutions such as Napster, Kazaa, and JXTA, whereas they utilize a set of servers or super-nodes to maintain the extra information about their sub-nodes or other super-nodes/servers. However, these mechanisms are not scalable, and they are vulnerable to attacks due to their centralization of indices in a small subset of the nodes. Along this way, the solutions based on distributed index nodes such as intelligent search [233], bloom filter based search [235], local indices based search [236], routing indices based search [237], dominating set based search [238] and adaptive probabilistic search [236] are more scalable and reliable.

4.3.2. Synchronous vs Asynchronous

We can model the resource discovery problem (specifically in a decentralized fashion) using concepts from graph theory, where the computing nodes and communication links, are represented by the graph vertexes and graph edges. At the initial state, some of the nodes may know about the resources of some other nodes, for example, each node possibly knows about a set of the original neighbors. Furthermore, each communication link between two nodes demonstrates that those nodes know about each other. These relations and the environment can be presented as a weakly connected graph (uninformed model). The resource discovery search algorithms can be defined as the solutions that can converge the above mentioned weakly connected graph to a complete graph, where all the computing nodes know about resources of other nodes in the network. In this way, to provide an efficient search algorithm, the network communication complexity, and the number of required rounds for graph conversion (in terms of time, hops or cycles), must be low. Based on the above abstraction, we can classify resource discovery algorithms in two categories: synchronous and asynchronous methods (see Figure 13), where this latter can be either informed or uninformed, deterministic or non-deterministic.

In synchronous methods [112, 239–242], the search algorithm proceeds synchronously, in parallel rounds, where each round is defined as the time required for each node in the network to communicate with one or more other nodes, learning about them and gathering information. Asynchronous methods [243–245] are based on the asynchronous communication model, where

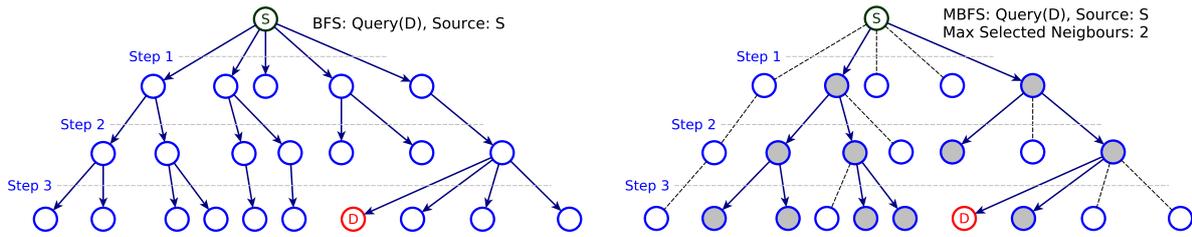


Figure 14: Examples of query processing using BFS and MBFS.

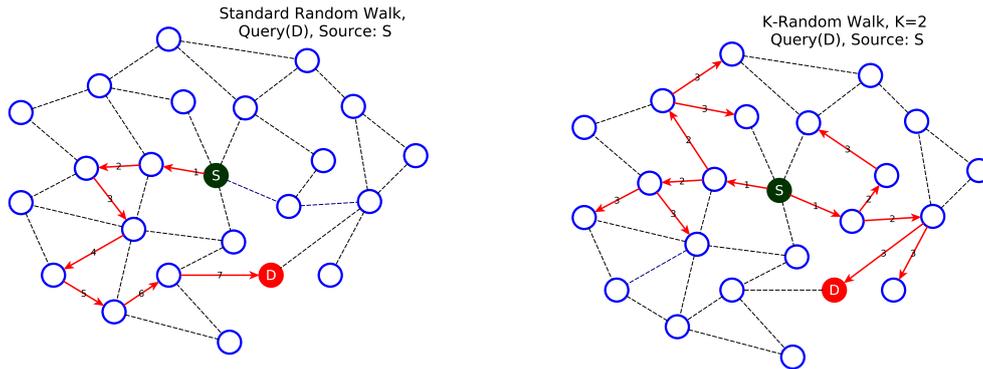


Figure 15: Examples of query processing using Standard Random Walk and K-Walkers Random Walk.

each node can send a message in arbitrary size to any of its neighbors in a way that the outgoing message eventually arrives in the destination node after an unbounded finite time. Moreover, unlike the synchronous method, there is no assumption to start the algorithm simultaneously on all the nodes, and the receiving nodes simply follow the First-Input First-Output (FIFO) model to serve the requesters.

Harchol et al., [112] discussed some well-known natural algorithms such as ALG-Flooding [246], Swamping [246, 247], Random Pointer Jump [246] and presented Name-Dropper [112] to perform a synchronous resource discovery. Table 5, compares these algorithms to other approaches on three performance measures: time complexity, pointer complexity, and message complexity. For a discovery request: the time complexity is the number of time steps taken; the pointer complexity is the

number of nodes/pointers (e.g., machine addresses) passed; and the message complexity is the number of messages sent. In the following, we discuss these algorithms in more detail.

When using the ALG-Flooding algorithm, each node only communicates with its (manually configured) initial set of neighboring nodes. Thus the newly added edges are not used for communication. In each round, every node sends updated information, including the new nodes that joined recently (see Figure 16). The number of required rounds to converge the graph to a complete graph depends on the diameter of the initial graph. Furthermore, in the ALG-Flooding algorithm, every pointer information must be transmitted over every edge in the initial graph. Thus, the network (or communication) complexity of the algorithm (concerning pointer and message complexity) depends on the number of edges in the initial graph. Using

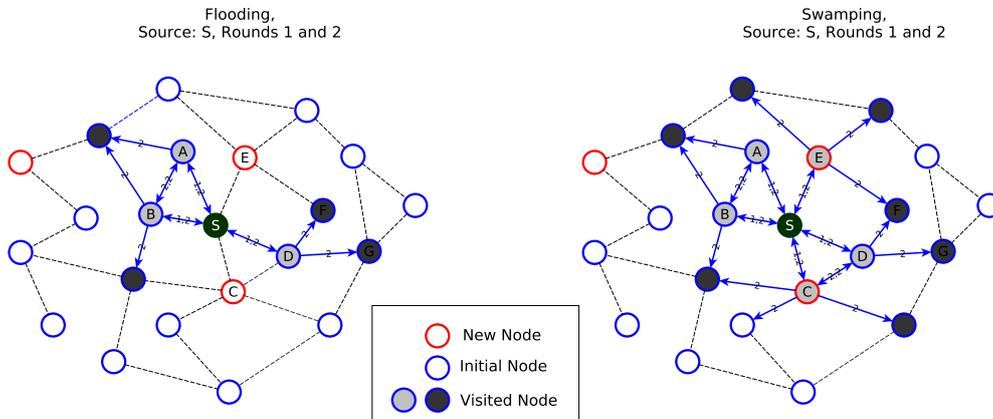


Figure 16: Examples of information dissemination using ALG-Flooding and Swamping.

Table 5: Comparison of some well-known synchronous search algorithms for state discovery.

Search Algorithm	Time Complexity	Communication Complexity	
		Pointer Complexity	Message Complexity
Absorption [248]	$O(\log n)$	$O(n^2), O(n^2 \log n)$	$O(n), O(n \log n)$
Kutten & Peleg [249]	$O(\log n \log^* n)$	$O(n^2 \log^2 n)$	$O(n \log n \log^* n)$
ALG-Flooding [112]	$d_{initial}$	$\Omega(n \cdot m_{initial})$	$\Omega(d_{initial} \cdot m_{initial})$
Swamping [112]	$O(\log n)$	$\Omega(n^3)$	$\Omega(n^2)$
Random Pointer Jump [112]	$\Omega(n)$ in worst case	$number.of.cycles \cdot m_{initial}$	$number.of.cycles \cdot m_{initial}$
Name-Dropper [112]	$O(\log^2 n)$	$O(n^2 \log^2 n)$	$O(n \log^2 n)$
Fast-Leader [250]	$O(\log^2 n)$	$O(n^2)$	$O(n \log^2 n)$

flooding-based methods for resource discovery in environments like large-scale Grids will reduce the overall system performance since flooding increases network traffic congestion.

Swamping is a flooding-based search algorithm with the difference that, each node may send requests to all of its neighbors, and not only a fixed initial set of the neighboring nodes. Since nodes transfer the current set neighbors to the target nodes using the request messages, the neighbor sets are dynamically changed and updated, which leads to a very fast search algorithm. However, this speed is achieved at the cost of wasting communication bandwidth, because many nodes will receive information about nodes that they already knew about (i.e., they have them in their local set of neighbors). This increases the network communication complexity very quickly, which reduces the algorithm's performance (see Figure 16).

Concerning the problem of the communication complexity of the Swamping algorithm, the Random Pointer Jump algorithm proposes that in each round, every node can only send a request to a single random neighbor. Upon receiving the request, the receiver node will send the information of its neighbors to the node that made the request (see Figure 17). The solution results in lower communication complexity (i.e., the average number of transacted messages and visited nodes to resolve a query will be decreased). However, the resource graph in the initial stage must be a strongly connected graph (informed model). Otherwise, it will never converge to a complete graph. Moreover, even for the strongly connected graph, there is a high probability that the number of rounds required to achieve a converged graph will be significant. The Random Pointer Jump with Back Edge [251] attempts to address this problem. Using the Back Edge algorithm, when a node receives a request, it adds the information of the sender to its neighboring set. Afterward, it sends its neighboring information to the requester node. This enhancement increases the performance of the Random Pointer Jump algorithm, but the main drawback is that there is still no guarantee that the resource graph converges after a specified number of rounds.

The Name Dropper is an enhancement to the Back Edge algorithm. In each round, each requester sends its neighboring information within the request to a single random neighbor. The receiver will merge the received neighbor information with its local set of neighboring information. The rest of processes will be performed just like the Random Pointer Jump with Back Edge algorithm (see Figure 17). Name Dropper achieves better performance by providing small communication complexity and a low number of rounds required. The drawbacks are that the algorithm is not deterministic and also it is not able to determine its termination point unless the number of nodes in the graph is

defined in advance. The Fast-Leader [250] algorithm has been proposed to overcome these drawbacks. The algorithm includes two steps. In the first phase, the initial strongly connected graph will be converged to a star graph, where a central node knows about all other nodes. This is achieved through a leader election algorithm. In the second step, the star graph will be converged to a complete graph by using broadcast propagation, where the central node broadcasts all the information to all other nodes. In opposition to Name-Dropper, Fast-Leader is a deterministic algorithm, which its runtime (number of rounds required to resolve a graph) and communication complexity match those of Name-Dropper. Furthermore, the convergence detection approach (i.e., recognizing the termination point of the algorithm, which is the time that the initial graph has converged) is an inherent feature of Fast-Leader, and the algorithm halts as soon as convergence is obtained. However, unlike Name-Dropper, Fast-Leader is only applicable to resolve the strongly-connected initial graphs. Another shortcoming is that the algorithm is more resource consuming regarding the memory usage and computation cycles per node in each round.

The above mentioned natural synchronous search methods have been used in many resource discovery approaches for different environments like Grids. However, these algorithms, in general, have some critical weaknesses. They do not support (or weakly support) dynamic environments considering different dynamicity issues concerning rapidly and frequently changing network topologies (i.e., node arrival, node departure, node failure, resource expiration, or adding new resource) and resource attributes (e.g., CPU load, memory utilization). For example, after running the algorithms (by converging the system resource graph to a complete graph), each node will have the information about all of the other nodes in the network. However, this would be useful only for static information, with the additional assumption that the system is static and reliable, which is far from real. In a different way, each node requires iterating the search procedure for updating its information, potentially resulting in inefficiency. Furthermore, using periodic updates also creates a huge amount of network overhead.

Essentially, in a real environment, it is not necessary that all nodes in the system simultaneously know about others, due to the fact that the transaction and maintenance of a significant amount of redundant information over the network is a waste of resources. For example, a resource discovery approach might be interested in only discovering the closest resources inside the border with a specific number of hops or latency.

4.3.3. Gossiping-Based Approaches

In gossiping, two random, non-requester members of a group repetitively talk (exchange information) about a query, generated by a requester in the system. Gossiping might be strongly related to epidemics, by which a disease is spread by infecting members of a group, which in turn can infect others. Epidemics refers to information dissemination between randomly selected members where the information (i.e., the “disease”) can be communicated among an expanding number of members [252].

In order to solve the problem of dynamic load balancing, Gossiping based search methods (Epidemic Query/Information Disseminations) provide alternative mechanisms to perform queries in the distributed environments whereas they have shown efficiency in information dissemination. They are based on flooding, but in a manner that generates bounded worst-case network loads. In general, to find a match for a resource request using a Gossiping protocol, the requester node starts to gossip about the required resource with some other randomly selected nodes (e.g., picks one of the neighboring nodes). Thus, the query (required resource) will be known for both of sender and receiver in the first round. Consequently, in the next rounds, each node that already knew about the query repeats the process by gossiping to another randomly selected neighbors (i.e., periodical querying/updating respectively by exploiting either push based or pull-based approaches) until the query is matched or the query is over aged. Nodes also gossip about the best match after they find the matches, (i.e., the best matches will spread through the network). Gossiping based solutions might be considered as a type of epidemic protocol because a node that gossips a query or a resource information can be seen as it infects its neighbor. The advantage of Gossiping is its robust capability for disseminating and delivering contents to (almost) all destination peers with a very high probability [227, 253, 254]. Thus these solutions may become fast, scalable and load balanced since they provide a statical guarantee on the number of nodes involved during the query/information dissemination process. Moreover, they are resilient to topology changes and churns due to their decentralized nature. The drawbacks come from the fact that Gossiping based search methods are more concentrated on the load bal-

anced query/information dissemination and they do not provide approaches in other aspects, such as the way to route queries, or mechanisms to match a query to a resource description. The works in [229, 255–258] are some examples of gossip based resource discovery protocols.

4.3.4. Bio-Inspired Approaches

Bio-inspired search methods (see Table 6) are based on biological systems, which have native capabilities to exhibit autonomy in various levels, ranging from molecular independent self-management and self-organization of organisms to the large-scale adaptation of animal in communities and colonies. Other than autonomy, these systems have some other desirable inherent properties such as scalability, adaptability, and robustness, which create motivation to apply bio-inspired search methods to discovery systems. A bio-system contains bio-organisms interacting in a bio-environment where bio-organisms are the self-organized autonomous entities without any central controller, and the bio-environment provides a medium for communication and interacting between bio-organisms.

There are a lot of similarities between the behavior and characteristics of the components of any distributed systems (e.g., P2P overlay) and the behavior of bio-organisms in the biological systems. Examples include birth (joining new member), death (node departure or failure), migration, replication, division, finding the food sources (resource discovery), chemical signaling (communication messages). As a specific example, for resource (food) discovery in biological systems, a bacteria can release a chemical signal that produces chemical gradients in the environments. Thus other bacteria in the vicinity can detect the gradients and know about the location of other signal-emitter bacteria. This mechanism can be used to disseminate the location of the source food, which has been already discovered by any one of the individual microorganisms. Another example is the mechanism that a colony of ants collaboratively use to find the food source, and build the shortest path to the colony.

Bio-inspired search algorithms for resource discovery have been studied and proposed in several research works. As example of this kind of search methods we can mention Tabu Search [280, 281], Ant Colony Algorithm [265, 266], Immune

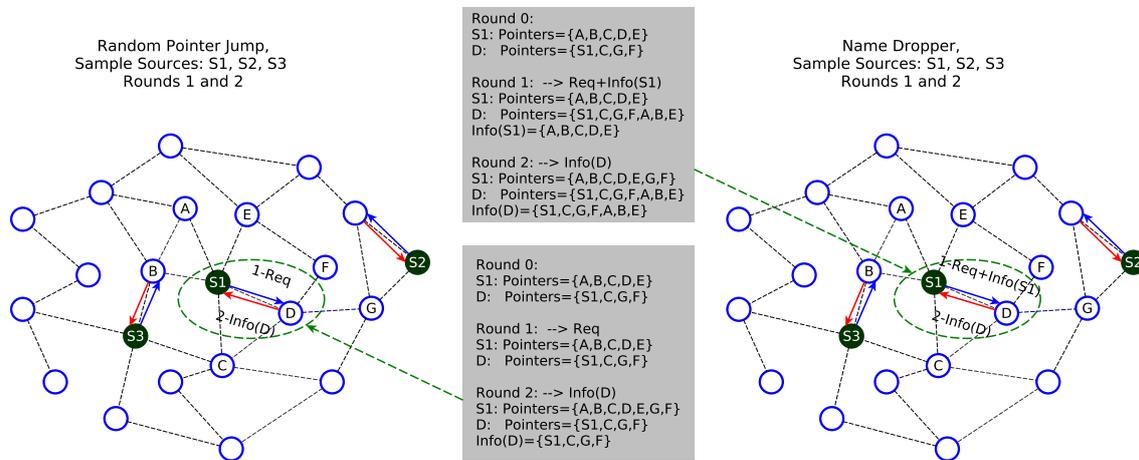


Figure 17: Examples of information dissemination using Random Pointer Jump and Name Dropper.

Table 6: Summary of bio-inspired search methods (SA: Search Algorithm).

SA	Description
Bee Colony [259–263]	A swarm-based optimization algorithm which is inspired by foraging mechanisms of honeybees. The scout bees stochastically search (global search) the new food sources, upon finding a new source, they will transfer the flower information by performing waggle or round dance. On the other hand, the worker bees evaluate the quality of the newly discovered flower sources through observing the dances and choose the elites (best sources) for foraging. Bees must avoid overcrowding and keep diversity, for this reason, they scatter in the proximity around the elite flowers (local search) while at the same time the scout bees start a new iteration of the global search. Bee Colony Algorithm (BCA) divides the search process into two steps, parallel implementation of the global search (parallel random search in variable space by scout bees) and the local search which is the local improvement of the current elites by worker bees. The elite selection mechanism in each iteration compares the quality of the discovered solutions from global search and local search with the quality of the elites in the last iteration.
Ant Colony [264–267]	Ant Colony Search is a meta-heuristic search method based on Ant Colony Optimization (ACO) which is inspired by the behavior of the real ants searching their environment to find the food sources. The ants (queries) start their search by randomly parallel scattering around the nest. The successful ants whose found the food sources (resources) will return to the nest using their memory and mark the (successful) path (between nest and food source) through emitting a chemical substance called pheromone on trails. The other ants, coming across the trail, follow the marked path instead of wandering randomly to check the food source. If they become successful to find the food, they will return the nest and reinforce the pheromone on the trail. For selecting a trail, each ant makes a local decision by comparing its experience with the environmental information (trails) which is updated by other ants and finally it selects the strongest path(trail) in terms of the density of pheromone considering the fact that the pheromone evaporates over time. In other words, in the intersections, the ants prefer to choose the strongest trail through comparing the already available trails marked and modified by different ants (i.e., indirect communication or stigmergy) instead of direct communication and exchanging information with other ants. Using this approach the pheromone of the paths with the long distance source will be more evaporated than the shorter tracks since for the long paths it takes more time for the ant to reach the nest. Thus, the density of the pheromone in the shorter paths would be higher than the longer paths. This leads to discovering the closest resources with higher probability. The ACO based search methods make benefits [266] for resource discovery in terms of autonomy (the nodes do not have any global information), parallel search, proximity-awareness (quick convergence to near optimal solution), efficiency (prevents a large-scale flat flooding [268]), flexibility (supporting multi-attribute range query [268]) and robustness (concerning system workload). There are several proposals for ACO based search methods such as Semant [269], NAS(Neighboring Ant Search) [270], ACS(Ant Colony System) [271] and Max–Min Ant System [272].
Neural Search [273]	Neural search is based on Artificial Neuron Network (ANN) [274] wherein a set of neurons (computing units, nodes or computing elements) are connected together to construct a network which mimics a biological neural network of the brain. Artificial neurons are the simple modelings of brain cells which are connected using both of feedback and forward links with different dynamic weights that create an adaptive system. The adaptive weights are the numerical parameters that are tuned by a learning algorithm during run time (or training/prediction phase), and conceptually they clarify the strength of the links which can be used for the link evaluation in the process of neighbor selection of the neural search. NeuroSearch [275] is an example Neural search which attempts to solve the overhead problem of the flooding based approaches such as BFS by selecting the best neighbor for query forwarding based on the individual evaluation of the output of the neural network (for a set of specific input parameters) for every one of the neighbors. ANN will create deterministic or probabilistic maps between input and out parameters which lead to specifying the weights for each one of neighbor nodes.
Viral Search [276–278]	Proposes a meta-heuristic search method based on viral infection using a biological analogy. It takes the advantages of Multi Agent Systems (MASs) [279] and combine it with some well-known approaches in Artificial Intelligence (AI). The viruses in the system are considered as part of the widespread infection, while each individual tries to make its benefits but leading in the overall benefit of the viral system. It is desirable for viruses to infect the individuals with the minimum level of health. The efficient viral infection can be achieved by continuously searching the individual microorganisms (e.g., bacteria) that are appropriate for infection (i.e., the best solutions which are the individuals with the less level of healthy). In this way, the viruses optimize their search results by infecting less healthy individuals (weak solutions). The probability to extend the domain of infection (in other words, achieving higher access to the new potential results) can be increased by systematically propagating the viruses which are lodged in unhealthy cells.

Search [276], Neural Search [273], Viral Search and Bee Colony Algorithm [259, 263, 282, 283]. Table 6 briefly describes some of these approaches.

4.4. Packet Propagation

Unicast is the most conventional communication method in resource discovery systems. The origin node (sender) explicitly addresses the receiver and sends any type of messages (e.g., query, reply, advertisement, etc.) directly to the destination node through the network. It is effective since the receiver address is clear and it does not require to occupy the network bandwidth with unnecessary extra communication messages. Also, the sender can know if the receiver already received the message. The problem is that in a pervasive distributed environment with dynamic resources, the destination nodes are not always known in advance. Thus it is required to employ other alternative propagation methods such as broadcasting, multicasting, publish/subscribe, manycasting and anycasting.

Broadcasting or simple flooding uses one to all communication mechanism to send packets to all the nodes in the network or subnet. Resource discovery systems employ broadcasting in the absence of predefined servers, which creates a lot of overhead in the network.

Multicasting for a number of nodes starts through establishing a multicast group by sending unicast initiate messages to each other. Once a node sends a message to a multicast address,

it causes several unicast messages from the sender to all the members of the multicast group. Multicast is used when the destination nodes are unknown at the source of the message at the network or link layer, and it is not clear which nodes are offering the requested resources for a particular query. This communication model, compared to broadcast, is still efficient. Publish/Subscribe [284–286] is similar to multicasting in nature; a receiver node subscribes certain services, which are offered by publisher nodes, and obtain these services directly or through intermediate nodes. Publishers constantly report all the service changes to their subscribers. They may also update intermediate nodes. Subscribers are not updated immediately, and they should fetch new data from the intermediate nodes as soon as they contact them, instead of direct updating.

Anycasting provides the capability to communicate between a source node and one member (the one closest to the source) of the anycast group (a group of target hosts). In other words, a single anycast request will get a single reply, which leads anycast to be considered as a strong way to make reliable scalable and transparent communications, with stateless distributed services such as resource discovery and DNS. For example, in DNS, a number of replicated DNS servers create an anycast group where they are listening to a shared anycast address. Upon getting a request, the DNS server with the shortest path to the requester will reply the request [287]. For the purpose of resource discovery, due to the inherent single-request/single-response (from the

nearest member) feature of the anycast communication model, and its capability for coarse-grained load balancing between the members of anycast group, it is useful as a transparent resource discovery primitive.

Manycast integrates the characteristics and advantages of multicast and anycast, leading to an efficient communication paradigm, wherein a source node sends the packets to a specified number of pre-defined members of a manycast group. In the following of this section, we mainly focus on anycasting since we consider anycasting as a powerful paradigm for managing and locating resources in decentralized distributed systems.

Despite the aforementioned strengths, anycasting has some limitations and weaknesses [288]. The first limitation is that session-based applications, such as all the implemented applications on top of the TCP layer can not benefit of the anycast addressing mode. This happens because it is possible that the subsequent packets from the same source node (and session) are routed to a different target host member of the anycast group (it has no knowledge of the TCP session state). The second problem is that anycast IP routing is inherently not scalable. In fact, the routes to different anycast groups in the routing tables cannot be aggregated. Widespread adoption causes a massive growth of IP routing tables, which reduce system scalability. If the members of an anycast group are scattered over the Internet, in each routing table, for each anycast group, it is required to have a distinct routing entry; this is costly due to the limited, efficient size of the routing tables. Moreover, the dynamic behavior of anycast members (joining and leaving members) leads to frequent changes of routing configurations (i.e., frequent changes in network topology), which possibly makes the system unstable, so that the routing and discovering protocols can not operate efficiently. There is another problem with anycasting that might affect its performance. The problem is due to the inherent static target member selection of anycasting which is based on the fixed indicator of the shortest path. In fact, it doesn't support multiple target selection constraints such as network congestion and current target load. Due to the above strengths and weaknesses, TCP-based services are not able to take advantages of anycasting.

However, anycasting, with its robust native capability to efficiently find nearby resources has been considered as an important communication paradigm to enhance the resource discovery approaches by leveraging the features and capacities, as mentioned earlier, through implementing an anycast based system. PIAS [173] and ASTAS [174] are two examples of anycast architecture which can be used to implement anycast based resource discovery systems.

PIAS is an IP anycast architecture, which makes use of a proxy overlay for advertising IP anycast addresses on behalf of group members and tunnels anycast packets to those members [173]. ASTAS is an architecture for scalable and transparent anycast services, which is based on PIAS. It establishes an overlay infrastructure compound by two types of nodes: Client Proxys (CPs) and Server Proxys (SPs). Both of these proxy nodes act as routers that advertise routes from their neighbor nodes to an anycast IP range into the routing substrate, by forcing IP packets containing the anycast members as the destination

address to pass through the overlay. Once a client node initiates a new session towards an anycast destination, the nearest CP registers the new session, and afterwards, it selects the most suitable SP forwarding the request to it. Upon receiving a new session request by a SP, it selects the most appropriate server node to process the request. Because of the stateful nature of proxy components, ASTAS provides more fine-grained and load balanced distribution of the service request over the available resources than PIAS.

4.5. Information Delivery

Individual nodes of a resource discovery system need to share information efficiently. Thus, the information delivery mechanisms must aim at generating little network overhead and bandwidth consumption, through the reduction of the volume and size of exchanged messages. For achieving this purpose, there are some well-known techniques such as caching, piggybacking and heartbeat checking.

Caching relies on storing successive resource advertisements of each node so that new queries for similar resources can be resolved locally. For example, when a resource matching a query is found in a node, that node will send the resource information backward to the original requester, which leads that the requester node and each of the intermediate nodes can cache the information. For the new queries, each node first checks its local cache to find the query match and, if local matchmaking is not feasible, the query will be forwarded to other nodes in the system. The system has to manage stale-information stored in the cache by either removing expired cache entries or by eliminating the oldest cache entries, to add new entries when the cache size limits are exceeded. Using caching mechanism can improve system efficiency (in terms of overhead, latency, bandwidth consumption and query workloads) by handling the queries locally instead of doing global query processing. Since consequent computing applications or application segments frequently require almost similar resources, caching the result of the previous queries might be useful to avoid repeating the same queries in the next discovery cycles.

Caching-based techniques have been widely deployed in many resource discovery research efforts. For example, some works [204, 258] propose an efficient resource discovery protocol based on proactive information caching, which supports the construction of self-structured overlays. They have an ant-inspired algorithm that uses selective flooding to exploit local caches, in order to decrease the number of explored nodes for each query. Caching mechanisms have been further improved through periodic exchanging of the cache contents between neighboring nodes, through an epidemic replication mechanism based on gossiping. Leveraging the caching mechanism enhances the system performance by reducing the number of hops to locate required resources and decreasing the overall discovery loads.

Piggybacking means responses can be included on top of acknowledgment messages. In other words, a regular reply message can carry (piggyback) extra information. Piggybacking might be costly in terms of the amount of information transacted in the system. However, it can significantly reduce the number of messages to resolve a query. For example, works include

[289–292] use piggybacking technique to facilitate information delivery. Unlike piggybacking, in heartbeat checking [293], a sender node periodically sends a signal (a very small amount of information) to other nodes, updating the status of the sender (e.g., in terms of availability). This mechanism is more compatible with the proactive discovery strategy, where a resource provider periodically advertises its resource information to the clients (e.g., subscribers).

It must be taken into account that the efficiency of an information delivery mechanism is largely dependent on its adaptability to the underlying computing environment. This is particularly important for highly dynamic computing environments. In such environments, it is significantly hard to maintain and analyze the reliability of the resources since available resources change rapidly, meaning that existing ones are removed or new ones are added frequently. Among the information delivery methods discussed above, heartbeat checking exhibits better adaptability to dynamic environments since it is based on real-time availability checking of resources. Caching-based methods may suffer from the unreliability of cached resource information due to rapid changes in the environment. Piggybacking may provide better efficiency compared to caching. However, piggybacking can impose extra loads on reply messages without guaranteeing the availability of resources.

4.6. Query Termination

In general, the resource discovery process starts by sending a query message to the network, where intermediate nodes/agents spread the query through different search and forwarding techniques (See Section 4.3). However, the query must be terminated at some point under some specific conditions. Otherwise, the discovery procedure would not be under control, resulting in the creation of extra unnecessary network overhead, without having a guarantee of finding the required resource. The following common situations describe how a query can be terminated, either by natural mechanisms or by a query termination method.

A) The resource required for the query is found in the current node, in which the query instance resides. Thus the query is successful, and the query response must be sent to the original requester either through direct communication or back-tracing.

B) The query already has visited all nodes in the system. Therefore, the query is stopped and deleted. The last visited node may report the query status to the originating node. The results inform the source node that the required resource does not exist in the system. Queries leave marks at each explored node, indicating that the query already visited this node. However, using blind searches (for example, in an unstructured system) it is a challenging issue to ensure that all nodes in the system have been explored by a particular query, since, the number of the nodes in the system as well as the network topology is unknown. On the other hand, even in a known system, a query instance may know how many and which nodes were visited by itself. However, it would be more complicated to know about other instances of the query, since communication and synchronization between several instances of the same query generate extra overhead.

C) The query is not able to be further forwarded to the next hop. Thus the query is stopped and deleted. This happens because of two possible reasons: First, the current node is a leaf node, which means that there are no more neighboring nodes to forward the query along, or the query was already sent to all neighbors. The second reason is that the query's TTL has expired.

Iterative Deepening is one of the termination techniques that can be used when the TTL expires while the query is not resolved, and perhaps the search space was not completely explored. If the originating node, after finishing an iteration (i.e., upon receiving the TTL expiration notifications from all the instances of the query) does not receive a query response regarding its required resource, then it starts a new iteration by resending the same query with increased TTL, to search within a larger radius of the search space. This process is repeated in the next iterations until either the query is resolved or the query's specific timeout is reached. The query timeout may be different from TTL, which is the number of visited hops by a query. The drawback of this solution is that, in each iteration, the query propagates through a substantially larger portion of the same nodes that have already been visited in the previous cycles, creating unnecessary search overhead.

Checking [234] is another termination mechanism which can solve the problems of Iterative Deepening. When a query-replica reaches a point that its TTL is expired, the node sends a specific checking packet to the original requester. Upon receiving the checking packet, the requester will see if the query already is resolved by other query instances. Otherwise, it sends back the checking packet including the information to increase the TTL of the query replica, and then the query proceeds being forwarded. Each query replica sends the checking packet either periodically after n hops or after TTL expiration. The drawback is the use of extra communication to exchange the checking packets.

Chasing Wave [294] is another possible approach. Upon receiving a query response from any one of the query replicas, the originating node sends chase packets to chase, and stops the other active query replicas. This mechanism works based on the concept that chasing packets traverse faster than query packets and it can be implemented by increasing the delay before forwarding, as the distance of the query replica to the originating node becomes longer. The paths to the active query replicas can be traced using their marks on the intermediate nodes, which indicate each query's next hop. Using this method may impose slower discovery process by increasing the delay before a query forwarded from far nodes.

Backward Synchronization is another alternative termination approach, triggered when a query-replica faces a deadlock, and the query is still not resolved. The deadlock might happen when there is no neighboring node (except the node where the query came from) to forward the query. It may also happen if all neighbors already have been visited by other query-replicas, or in the case that the query's TTL is expired and we do not want to extend the search diameter to discover far distance resources. In such a situation, the current node sends out the query backward to its parent node (the previous node that sent the query

replica) while it decreases the TTL of the backwarded query. The backward query message will proceed to the parent node in the normal way (for example, by forwarding the query to a randomly chosen neighboring node which has not been visited by other query replicas). In any stage, if the query faces a new deadlock, the backwarding is replicated unless the query is resolved or the originating node becomes the parent node. Like the Chasing Wave approach, queries leave marks on visited nodes indicating their next hop. This information is used for backwardness and determining the visited neighbors by other query replicas. This Backward Synchronization method can be used for search methods like Random Walk. Using this technique (along with flooding approaches) might not be efficient because, in flooding-based search methods, all the possible behind paths have already been explored, leveraging a large enough number of query replicas.

5. Evaluation Aspects

In the current literature, there are a set of major challenging issues for resource discovery in large-scale distributed computing systems including scalability, efficiency, reliability, flexibility, and heterogeneity. In this section, we discuss these issues, and accordingly, we provide a set of evaluation aspects which must be considered in the evaluation of any discovery approach. Table 7 provides an overview of the most important performance factors (i.e., evaluation metrics) for assessing discovery protocols [295–299]. In the remaining of this section, we further discuss these issues and relevant evaluation strategies to measure the performance factors.

5.1. Efficiency

The efficiency of a resource discovery solution can mostly be represented and demonstrated along the following conceptual metrics and functionalities.

Discovery Latency: One of the most important performance metrics for resource discovery protocols is the discovery response time or the end-user latency. The discovery latency in distributed systems might come from different sources, which are mostly related to the network transfer times, query processing algorithms, type of querying (e.g., the number of dimensions in multidimensional querying), network size, computing environment and the amount of parallelism (e.g., the number of walkers). In the case of network transfer times, the possible efforts to minimize latency is to reduce the number of bytes sent and also the number of times they are sent. Thus, the discovery mechanisms and algorithms are required to be optimized to transact the discovery messages (e.g., requests, replies, updates, etc.) with regards to minimizing network bandwidth. In this way, metrics such as the discovery load (i.e., the average number of transacted messages per query), the number of hops per query, and the number of explored (visited) nodes/resources per query can be used.

Load Balance: During a discovery procedure, the query/discovery load will be distributed among all potential resource information providers. Employing a load balanced technique for

querying will enhance the total scalability of the system. Furthermore, since resource information providers are completely distributed in the system, and querying mechanism follows a symmetrical distribution, it also prevents any centralization, bottlenecks and single points of failures.

5.2. Scalability

Scalability for resource discovery represents the ability to cope with variations in the system effectiveness. The effectiveness can be measured in terms of the system’s performance, throughput and QoS seen by the resource discovery clients, while a set of critical system parameters are changing (e.g., the system size is increasing) (see Figure 18). The discovery clients can be defined as end users (or job schedulers) in Grid or Cloud, process managers in distributed operating systems, and applications in HPC.

It must be taken into account that scalability is one of the most important key factors to evaluate distributed systems in large-scale environments. Referring to the nature of large-scale computing in LDCE, the system size is the most significant changing factor for scalability evaluation in the majority of the resource discovery research works (other changing parameters, such as query types and query dimension, are occasionally considered for study). Increasing the system size has an impact on the system performance by creating larger communication overhead, which increases network latency and results in an increase of the discovery response time (or discovery latency). Thus, the communication cost can be measured concerning the number of transacted messages among the distributed nodes during query processing for a particular application resource request.

The ability of the system to grow in terms of available resources/peers, such as cores/nodes, without impacting overall system performance will impose conditions/restrictions in various aspects such as system architecture, overlay construction, communication model and query processing. In the remaining of this section, we address these aspects.

In order to avoid any central point of failures and bottlenecks, discovery systems ideally should avoid relying on centralized architectures. However, implementing an efficient fully distributed design is not attainable or has drawbacks as well. Therefore, depending on the case, a hybrid architecture such as a distributed hierarchy might be useful. We must note that in a distributed system it’s impractical to have a global view of the system while making it scalable, instead it is possible to make specific queries to the system, but it leads to a more limited view of the system. Furthermore, the distributed architecture must be loosely coupled between nodes. The advantage is that nodes are not affected by failures in other nodes.

The ideal system behavior would be for the communication overhead (number of exchanged messages) or query response time (as the result of decreasing overhead and latency) to stay constant as the number of nodes/resources (e.g., processors) increase. In a system that is not scalable we will expect that the system eventually reaches a point where the discovery latency/overhead grows significantly worse with the system size. Due to the aforementioned impacts on the scalability of the other

Table 7: Overview of the evaluation methods for resource discovery.

PF	Evaluation Methods	Description
Scalability	Quantitative Assessment (e.g., Overhead, Discovery Latency, Bandwidth Consumption, Query Load)	The capability of the resource discovery protocol to maintain its performance regardless of the other system parameters such as network size, query dimension, and environment dynamicity. For example, in a large many core system, we must clarify whether the discovery procedure is system size independent in the case of performance, throughput, and overload or not. Besides, the discovery protocol must scale up and scale out efficiently across multiple/many cores, multiple/many processors.
Efficiency	Quantitative Assessment (e.g., Network Overhead, Discovery Latency, Bandwidth Consumption, System Utilization)	The ability of the system to be cost efficient concerning computation and communication cost in all of three discovery stages: establishing the system, performing resource discovery and system maintenance. (e.g., the discovery must be fast as much as possible). The ability of the system to maximize the utilization of all the potential resources (e.g., idle cycles).
Reliability	Quantitative Assessment (e.g., The rate of successful/unsuccessful querying, Recovery Time)	The ability of the system to prevent and manage faults. (e.g., unavailability of peers/resources) The discovery results must be accurate. (e.g., non-deterministic search results)
Flexibility	Qualitative Assessment	The ability of the system to perform different type of Complex querying such as Multi-Dimensional, Range, Multi-Dimensional Range, Aggregate, Nearest Neighbor, Exact, Partial and Keyword Querying.
Dynamicity	Qualitative Assessment	The ability of the system to cover different kinds of dynamicity. (e.g., dynamic attribute values of resources, frequent arrival, departure and failure of the nodes)
Heterogeneity	Qualitative Assessment	The ability of the system to work in the environment with different types of heterogeneous resources. (e.g., Low-Heterogeneity: resources with different attribute values, High-Heterogeneity: resources with different set of attributes)
Grouping	Qualitative Assessment	The ability of the system to establish a self-organized overlay or platform which can support some valuable inherent features such as proximity-awareness, semantic-awareness, and QoS-awareness
Autonomy	Qualitative Assessment	The ability of the system to be purely decentralized. The autonomy of the peers to local control and manage their data and behavior.

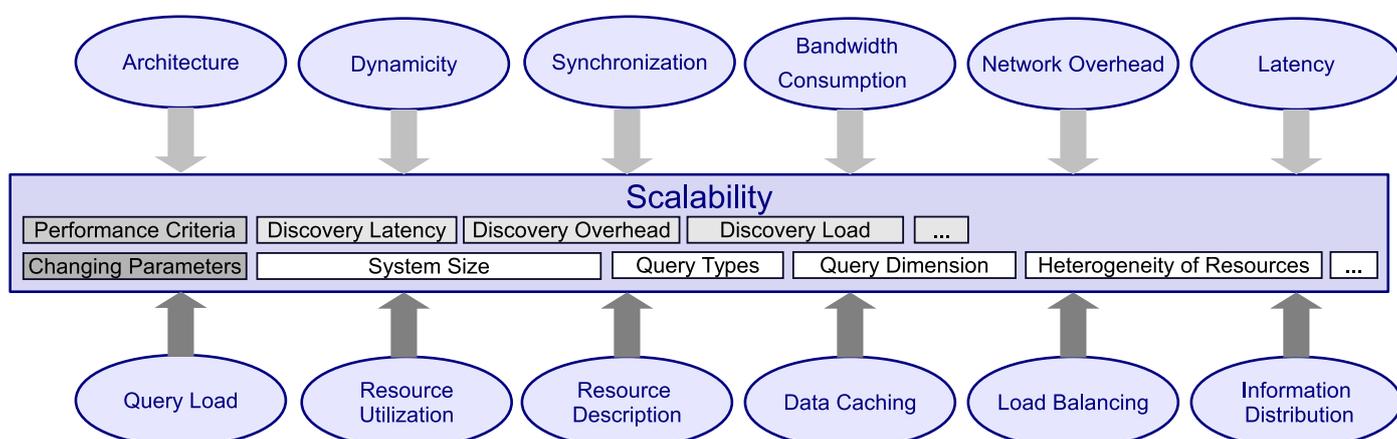


Figure 18: The important elements which have impact on the scalability of a resource discovery system.

important system factors (such as performance and resource utilization), it is possible to evaluate the scalability of the discovery system by studying and analyzing the system effectiveness.

In order to create a highly scalable system, discovery systems should support dynamic behavior and dynamic characteristics of resources, while tolerating failures and support load balance. Furthermore, the components of the discovery system should communicate asynchronously and efficiently, i.e., avoid high throughput in communication. Asynchronous querying gives freedom and autonomy to the peers or enables discovery components to have independent control over their data and behavior regardless of the global system structure, which increase the level of system decentralization. In other words, we can measure the system scalability based on the extent of its distance from a fully centralized system.

When system size increases, resource discovery must enhance resource utilization by providing support to make an efficient use of the increasing available resources, while avoiding overhead. It must perform load balancing based on resource demands while it avoids excessive communication overhead in terms of latency, query workload per each node, memory access, synchronization

and also traffic congestion on the communication routes within the system. We must also take into account that some amount of synchronization and message transaction are potentially required to maintain the discovery system (e.g., recovering from failure, handling departure of the nodes or arrival of new nodes)

The problem of scalability comes from different sources. For example, communication between a large number of resources will increase communication overhead, latency, jitter and causes packet loss, which results in a scalability problem. Some other significant sources are memory accesses, synchronization, and signaling. Frequent remote memory accesses, because of data dependencies or memory constraints, increase memory access latency. Data or state synchronization generates communication traffic, and finally signaling overhead of communication protocols (state maintenance) has scalability concerns. In a scalable system, it is expected that the system performance (e.g., in aspects like discovery latency or overhead regarding storage, communication, and computational load) is independent of system size and the resource discovery scales with increased system resources. We can evaluate the scalability of the discovery system by assessing the impact of changing the parameters of the

computing environment such as network size in the aforementioned scalability issues. Table 8 provides some examples of the evaluation of scalability methods in current resource discovery literature.

5.3. Reliability

Reliability is one of the major concern for resource discovery particularly in a HPC environment, which requires reliable, fast execution of the tasks on computing resources. We can distinguish between different types of reliability for resource discovery concerning the following aspects:

Accuracy (also known as Correctness): It concerns the accurate description and representation of queries and resources. In other words, accuracy is how efficiently the query/resource description model of the resource discovery protocol represents the features and complexities of the real world applications and hardware. For example, a simple flat attribute-based resource description model is not able to precisely demonstrate the characteristics and behavior of the hardware processing components such as memory hierarchy, manycore architecture and interconnection networks in a large-scale HPC environment. Thus, the resulting resource discovery approach might not be reliable in such environment. Additionally, in another example, providing a query description model that only supports exact match, single resource querying, would not be able to satisfy the resource requirements of a parallel multi-threaded application, which simultaneously needs to discover a group of connected heterogeneous resources with particular QoS communication values along each connected edge. In fact, the accuracy evaluation of the query/resource description model is part of the resource discovery assessment, which partially specifies the reliability of the discovery.

Reliability also concerns an accurate result of resource discovery procedure (i.e., returning results that precisely meet the requirements of the resource requester). It means that the discovered resources must carefully satisfy all of the query conditions. In this way, the success rate and hit rate are two important criteria for evaluating the reliability of discovery approaches. These metrics, as well as cost per query and cost per hit, can also be conducted to assess the system efficiency [258, 302–307]. A query is considered to be successful when at least one existing resource, matching the query, is detected. We note that a resource discovery solution is known to be deterministic if any query with an empty result can assure that even one resource matching that query is not existing in the whole system. Thus we can conclude that a deterministic resource discovery process provides reliability at least in terms of accuracy and success rate. Similarly, the hit rate (i.e., recall rate, success rate) measures the percentage of successfully discovered resources out of all matching ones (see Equation 1).

$$\text{Hit Rate} = 100 \times \frac{\#\text{Hits or \#Discovered Resources}}{\#\text{Matching Resources}} \quad (1)$$

Dynamicity and Fault Tolerance: Reliability must cover the dynamicity issues. The resource dynamicity means that any

node/resource in the system can join, leave or fail at any time (i.e., dynamicity in terms of topology change as well as resource life-cycle) and also it can change its characteristics (i.e., dynamicity in resource conditions and availability) [308, 309]. Thus, resource discovery performance must be evaluated using different configurations and dynamicity parameters (such as churn rate which is the percentage of nodes/resources in a given time frame that fails, leaves or joins the system) for the environment. In fact, in an unreliable system, it might be possible for a requester to get a resource discovery result containing information about a resource/node that already has left the network.

Resource discovery solutions must have mechanisms to deal with failures. These come from different sources that are mostly related to centralized discovery architectures (i.e., single point of failure and bottlenecks), dynamicity issues (i.e., dynamic attributes, node/resource joins, leaves and failures) and churn conditions. Tolerance to failures and churn is expected to be an essential characteristic of discovery systems particularly in very high dynamic environments containing a large number of non-dedicated resources. Similar to the resource discovery evaluation (in terms of supporting dynamicity), the discovery system must be evaluated for different configurations range from very stable to very volatile environments to prove the system toleration in different levels of churn that are to be expected from non-dedicated resources [310, 311]. There are several research works in the current literature [312–314], which analyze the impact of churn on the discovery performance. As examples of these works, [315, 316] concludes that the DHT based discovery solutions cannot efficiently cope with high churn rates. Rather, they may generate some non-tolerated failures like unexpected time-outs (caused by a finger pointing to a departed node) or lookup failures (caused by nodes that temporarily point to the wrong successor during churn) [317]. ECHO is a recent work which copes with DHT churn problems through implementing a tree-based index structure on top of DHT overlays [103].

Resource Reservation and QoS: Resource reservation capability is required for resource providers to reserve resources for particular applications (like multimedia applications). This means that resources are only freed when their associated processes are done. In other words, resource reservation ensures that the resources required are available for the processes upon the need.

Validity: Each resource can have an expiration time when after a given period of time, the resource becomes unavailable. It might happen that the discovery results contain an invalid resource which has already been expired.

5.4. Flexibility

Flexibility refers to the ability of the discovery system to express the capacity of resources and query requirements, both provided and sought, in a form that is convenient for the resource discovery users. The discovery mechanism must allow querying with very specific and detailed conditions (i.e., complex querying) as well as loose ones (general querying). Here, we elaborate the most important querying functionalities, which represent various aspect of flexibility for resource discovery solutions.

Table 8: Some examples of the evaluation of scalability methods in other resource discovery research works (RD: Resource Discovery, ET: Evaluation Tools, PL: Planet Lab, BM: Berkeley Millennium, IF: Iamnitchi & Foster, SG: StarGrid, MT: MatchTree, OS: OntoSum, NA: Not Available).

RD	Measured Criterias	Changing Parameters	Scalability Evaluation Results	ET
Ganglia [45]	Performance overheads , average bandwidth consumption per node for monitoring in a single cluster, total bandwidth for aggregating data from a set of clusters	The number of nodes within a cluster and the number of sites being federated	Linear scaling in packet rates, Linear scaling in local-area, Bandwidth consumed as a function of cluster size	BM, PL
OS [192]	The metric of recall rate, which is defined as the number of results returned divided by the number of results actually available in the network	Number of nodes in the network	OntoSum’s recall decreases less with the increase in network size	NA
MT [48]	Query response time, Bandwidth consumption, Query completeness	Query Types (easy and difficult queries)	A hierarchical result aggregation provides balanced processing overheads among resources with scalability. Several heuristics are proposed to improve the query response time and to decrease overall network overheads, and they are evaluated through large-scale simulations. It doesn’t present any direct evaluation method to evaluate the scalability	PL , OverSim
SG [300]	Throughput and query response time, Throughput is the average number of queries processed by the resource discovery component per second	The size of the Grid, number of nodes, number of concurrent requesters	The results prove the scalability for the limited size Grids	NA
IF [301]	Response time as the number of hops traversed for answering a request, success rate, which is the percentage of successful queries considering the dropped requests because of dead ends or exceeded TTL	The number and the frequency of shared resources/nodes, Different values of TTL, Under different sharing and usage policies (different number of existed resources for each particular resource type)	Evaluation on the top of the proposed resource discovery architecture shows that the relative performance of the different forwarding algorithms considered is independent of the average resource frequency. The resource discovery performance is correlated to the sharing characteristics.	Testbed

Employing these significant querying features enriches the resource discovery module to be more powerful and applicable to fulfill the requirements of various applications, configurations, and computing environments.

Multi-Dimensional and Range Querying is the capability of the discovery system to process queries containing multiple attributes either dynamic or static in specific ranges of values. The discovery results must satisfy every one of the conditions for each particular attribute. This would be more complicated especially when each single attribute is required to be qualified within a specific range of values. Since the resource contentions along multiple dimensions might happen in the environments with the uncoordinated analogous queries, the problem of multi-dimensional range-querying is known to be a challenging issue. Also, the multidimensional querying will probably reduce the rate of resource matching, while increasing query latency and bandwidth consumption.

The impact of multi-dimensional range-querying can be evaluated by using evaluation factors such as Failed Task Ratio (F-Ratio) and Throughput Ratio (T-Ratio) [318] which directly reflect the rate of resource matching for any resource discovery protocol. F-Ratio is the ratio of the of the number of failed (unsuccessful) tasks (i.e., the tasks that cannot discover any qualified resources) to the total number of generated tasks in a particular time period. T-Ratio also refers to the ratio of the number of completed (executed) tasks to the total number of generated tasks within a specific time period.

In the current literature, there are several resource discovery solutions that support multi-dimensional, range and multi-dimensional-range querying through leveraging several techniques. As examples of these approaches we can mention CAN-based protocols (e.g., CAN [61, 319, 320], RT-CAN [321], PID-CAN [322]), MAAN [323], Mercury [66, 324], Armada [325], Murk [326], Skip-Tree [327], SWORD [328], Node-Wiz [108] and MatchTree [48]. Moreover, a comprehensive survey on multi-dimensional methods and techniques can be found

in [329].

Resource Graph Discovery is the capability to discover a graph of resources considering both individual characteristics and interconnecting properties of resources. For example, in order to allocate resources for the threads in an application graph, we must consider the communication conditions and limitations among the application segments. Therefore, the classical approaches of resource discovery, which are mostly relied on finding resources according to the individual resource characteristics cannot be efficient. In fact, resource graph discovery is the process to locate and select the best possible matched graph of resources in the system where all the edges and vertexes can satisfy the query conditions.

Aggregate Querying generally relies on tree-structures to aggregate and combine results from a large number of nodes in a hierarchy. The typical examples of aggregation queries are Average, Median, Count, Sum, Maximum, Minimum and Top-K [330–334].

Nearest Neighbor Querying (or k Nearest Neighbor Query - kNNQ) means that for each given query, the discovery system must be able to return the k -Array = $\{r_1, r_2, \dots, r_k\}$ of results, in which r_i is the i -th nearest qualified resource of the requester. In other words, the resource discovery provides the list of discovered resources considering the priority of the closer neighbors.

Exact Matching is one of the general search functionalities which can be supported in most of the resource discovery systems. However, structured overlays such as distributed hash tables, provide better support for exact matching, in comparison to the other systems, through forwarding the query to the node which values of its keys precisely match the query’s requirements. In this kind of querying, typically, the query looking for “the resource whose key is K ,” due to the overlay mechanisms will be forwarded to the peer which is responsible for the address

hash(K). Thus, the querying operation results in a forwarding operation, and its efficiency would be directly dependent on the forwarding efficiency on the overlay.

Partial Matching and Keyword Search, are challenging for structured discovery systems. In many querying cases, for a given exact matching query, the results might be empty while in the system, there still exist resources that can partially meet the query conditions. i.e., for most applications, even the partially matched results (depending on their matching degrees) can be acceptable and provide benefits to requesters.

In keyword searching, users ask for any resource containing a given set of keywords advertised in (or extracted from) its meta-data or its resource description. Structured overlays such as pure DHT based discovery systems (e.g., CAN, Chord and Pastry) have addressed several of the issues related to the reliability and scalability that plagued the traditional P2P overlays (e.g., Napster and Gnutella). However, the advantageous functionalities such as keyword searching and partial matching presented in Napster and Gnutella are missing in DHTs that aim to replace them [335]. On the contrary, in unstructured systems like GIA [336], these types of querying are easily supported, basically because indices are mostly local and for a query visiting a peer, it would be easy to lookup the index table for a resource having a set of required keywords.

In order to support keyword search in structured systems, there are some typical approaches such as Inverted Indices [309, 337] and Bloom Filters [305]. The first method consists in the creation of inverted indices where, for each keyword k , the inverted index maintains the pageranks. The pagerank value specifies the priority of a resource for a particular keyword with regards to other resources, i.e., it is used to order indices at any resource. Besides, the inverted index stores the pointers to all of the resources/nodes in the system which announce that keyword in their descriptions. Thus, a potential approach is to record the inverted index for the keyword k at the node which is responsible for address hash(k). The underlying mechanism to answer a keyword query, containing a set of keywords $K = \{k_1, k_2, \dots, k_n\}$, in most of the optimized DHT-based discovery systems [338] such as Kademlia [339–341], includes two steps, where in the first step all inverted indices relative to each keyword k_i , $\{i = 1, \dots, n\}$ will be retrieved, and in the last step, these indices will be intersected to achieve a set of resources that contain all the keywords in their resource description or meta-data. The inverted indices guarantee to provide optimal hit rate, but they potentially can generate a huge amount of network overhead particularly when a large number of resources present a required keyword in common [342]. This happens because the inverted indices retrieved have to be sent to a single node in the network, which is responsible to perform intersection.

Bloom Filters (BFs) [9, 235, 343, 344] are another solution to perform keyword searching in structured systems. A bloom filter can be defined as a hash-based data structure which summarizes membership in a set. Thus, for the purpose of intersection between the keyword sets of the individual resources/peers in the system, it is possible to transfer a BF of a set instead of sending the set itself. This significantly reduces the amount of

communication required for supporting keyword searching in the system.

Other than Inverted Indices and Bloom Filter, there are some other well-known supportive techniques such as replication, partitioning, caching, ranking and incremental results [335], which facilitate performing the keyword searching on top of structured overlays. Several works [305, 314] provide surveys on the various aspects of the search methods containing keyword lookup, range queries, multi-attribute queries and aggregation queries.

5.5. Heterogeneity

The heterogeneity of a resource discovery solution can be evaluated along the following different aspects:

Various Administration Domains: The peers and resources can potentially belong to different underlying administrative domains and sub domains, particularly in large-scale, geographically distributed computing environments (e.g., Grid). Evaluation of resource discovery approaches on top of the real test-bed environments such as PlanetLab and OneLab might be a proper option to evaluate heterogeneity in terms of various administration domains. PlanetLab is a global research network consisting hundreds of nodes, geographically distributed in different sites belonging to different academic institutions and industrial research labs.

Various Hardware, Software, and Infrastructures: The resource discovery approach can be adapted to work simultaneously in different computing environments with various types of resources in terms of hardware (i.e., various hardware infrastructures) and software platforms (i.e., heterogeneity of the runtime environments, network protocols, operating systems, different data representations, data models, data structures and data accessing policies). In fact, resource discovery can be evaluated in the presence of heterogeneous computing cores (i.e., heterogeneity of the resources in a cluster or even inside the same system and chip, multicore and manycore processors, CPU, GPU, Field Programmable Gate Array (FPGA), Cell Broadband Engine (Cell BE), GPUs mixed with CPUs), heterogeneous hardware implementations (i.e., various architectures and memory hierarchies), heterogeneous Network on Chip (NoC) latencies (i.e., various networks and interconnections, heterogeneity on the Internet). As an example, the resource discovery is required to find heterogeneous resources for the purpose of code adaptation in HPC and cluster computing through adapting the machine code on the fly to different hardware platforms and operating systems, which is one of the most difficult scenarios considering heterogeneity.

Various Applications: A resource discovery approach can be created for different purposes. Thus, due to the resource discovery objectives, it can be used for different type of users, system components or applications. For example, a resource discovery protocol which is designed for the purpose of distributing task execution and task migration in computing environments such as Grid, Cloud or HPC must provide the requirements for heterogeneous applications such as HPC and Real-Time applications. HPC applications have to make use of the vast amount of parallel resources. This requires that the HPC application itself can exploit as much as concurrency as possible. In other words, HPC

Table 9: Summary of comparison between some of the well-known resource discovery approaches for different evaluation factors.

Approach	Resource Description	Efficiency	Scalability
SWORD [328, 345–347]	Native SWORD XML syntax or a Condor ClassAd.	The load balancing mechanism in SWORD is less efficient in an environment with a non-uniform distribution of peer ranges. SWORD is a DHT-base approach, and it uses multiple overlays.	++
Node-Wiz [108, 109]	It uses a relational model and standard SQL query processing in Node-Wiz-R.	Load balanced approach. Supports clustering and self-organization of the overlay. Uses single distributed indexing mechanism.	++
MDS-4 [43, 44]	It uses an extensible resource specification language based on Web Services Resource Framework.	Uses LDAP to create the overlay. Its overlay is based on GIIS and GRIS.	++
MatchTree [48]	Leverages Condor’s condor-startd daemon which provides summarized system monitoring metrics while it provides resource information in a ClassAd format for matchmaking. It is developed on top of Brunet P2P overlay.	Reduces query response times through redundant query topologies, dynamic timeout policies, and sub-region queries. Leverages a self-organizing recursive-partitioning multicast tree for query distribution and result aggregation. Balanced processing overheads among resources.	+++
CycloidGrid [50]	A set of attributes describes each resource in the system. These attributes are CPU speed, the amount of RAM, available hard disk size, operating system, and processor model. A decision tree (DT) does the classification of resource attributes in this architecture	Improves the response time of user’s requests. Distributes the load between peers based on QoS constraints of requests, round trip time (RTT), and the current load of resources.	++
OntoSum [192]	Organizes a Grid network by a semantically linked overlay representing the semantic relationships between Grid participants. Proposes a lightweight indexing summarization scheme based on Triple Filter which is an extension of the classical Bloom Filter data structure.	Uses a semantics-aware topology construction method which significantly improves the discovery efficiency in Grids through propagating the discovery requests only between semantically related nodes. Provides the ability to locate semantically related results. In OntoSum, Grid nodes automatically organize themselves based on their semantic properties to form a semantically-linked overlay network.	+++

applications are highly concerned with scalability constraints to support high degrees of concurrency [348]. But, Real-Time applications are more concerned with timing issues. They are not aiming to increase the level of parallelization. Rather, they are specifically interested in handling their timing constraints which bear a resemblance to the synchronization estimation (i.e., execution timing) in distributed applications. In other words, by exploiting the timing constraints applicable in real-time scenarios, the distributed synchronization behavior can be improved further.

6. Comparison

As we discussed in previous sections, there are a set of essential functionalities, features, and performance indicators for resource discovery solutions to meet all the major requirements of different applications in various computing environments. In this way, we have elaborated the most significant resource discovery evaluation factors and criteria concerning Scalability, Efficiency, Reliability, Flexibility, and Heterogeneity. In this section, we select three major groups (in terms of popularity) of resource discovery solutions for distributed computing environments: Grid-based (e.g., MDS-4, OntoSum), P2P-based (e.g., SWORD, MatchTree) and Hybrid-Approaches (e.g., NodeWiz, CycloidGrid).

Moreover, we choose some of the well-known approaches as representative of each group, and we compare them based on the evaluation factors that are discussed in the previous sections. Table 9, 10 and 11 provide a summary of comparison between the aforementioned resource discovery solutions considering the qualitative assessment through different evaluation factors.

As shown in Table 9, Grid and P2P based approaches (such as OntoSum and MatchTree) provide better scalability compared to Hybrid approaches. Both OntoSum and MatchTree are decentralized solutions. As we discussed previously, decentralized approaches naturally have potential to provide a high level of

scalability. However, the amount of communications required for decentralized discovery may have an impact on the scalability. Of course, there are also techniques in the literature, as we discussed before in Section 4.3, for reducing the discovery communication overhead. In this way, OntoSum decreases the discovery cost by propagating the discovery requests only between semantically related nodes, meaning that only a subset of nodes is explored instead of all nodes. MatchTree also decreases the discovery cost by applying a mixture of query termination (e.g., dynamic timeout policies), query partitioning (e.g., sub-queries and multicasting on sub-regions) and query guidance techniques (e.g., dynamic detection of redundant queries).

From Table 10, we see that P2P based approaches such as SWORD and MachTree are the most flexible resource discovery solutions among those analyzed. But their capability to deal with the heterogeneity (of resources) is lower than that of Grid and Hybrid approaches. SWORD and MachTree support complex querying including multi-attribute matching, range queries, resource graph discovery and also string/regular-expression matching. These features increase the flexibility of querying, enabling to perform several different types of querying. However, for both approaches, the level of detailed conditions, that can be expressed for each query, is low. In fact, they only support task-level querying which is quite coarse-grained mechanism for dealing with the heterogeneity of resources in computing environments.

Furthermore, in Table 11, it can be seen that P2P and Hybrid based methods enable more sophisticated features and functionalities compared to Grid-based solutions. The reason is that both P2P and Hybrid discovery approaches can inherit some natural features of P2P strategies (such as load balance, fault tolerance, and self-organization). This enriches P2P based approaches in terms of features and functionalities compared to non-P2P based approaches.

Table 10: Summary of comparison between some of the well-known resource discovery approaches for different evaluation factors.

Approach	Reliability	Flexibility	Heterogeneity
SWORD [328, 345–347]	Dynamicity in terms of Dynamic Attributes and Dynamic Topology is supported.	Supported in terms of Range Query, Multi-Dimensional Querying (using multi-query approach), Aggregate Query(Resource Graph Discovery) and Nearest Neighbor Querying	++
Node-Wiz [108, 109]	Dynamicity in terms of Dynamic Attributes and Dynamic Topology is supported. Quality of Service is supported.	Supported in terms of Range Query and Multi-Dimensional Querying	++
MDS-4 [43, 44]	Resource Reservation is supported. Fault Tolerance is supported. Dynamicity in terms of Dynamic Attributes and Dynamic Topology is supported. Quality of Service is supported.	Not Supported	++
MatchTree [48]	Guarantees query completeness. Fault Tolerance for the failures such as node churn, internal node failure, and consecutive packet drops is supported. Dynamicity in term of adding a new attribute is supported. Serves a different quality-of-service to users with distinct demands.	Supports complex querying including multi-attribute matching, range queries, string and regular expression matching	+
CycloidGrid [50]	Quality of Service is supported.	Not Supported	+++
OntoSum [192]	Supports accuracy in term of hit rate (or recall rate). Supports dynamicity in terms of probability to issue a query, probability to leave the system and the probability of new nodes with new resources joining the system.	Supported in term of neighbor discovery query	++

Table 11: Comparison between some of the well-known resource discovery approaches for different evaluation factors.

Functionalities	SWORD	Node-Wiz	MDS-4	MatchTree	CycloidGrid	OntoSum
Load Balance		✓		✓	✓	
Fault Tolerance		✓	✓	✓		
Self-Organization	✓	✓		✓		✓
Range Query	✓	✓		✓		
Multi-Dimensional Query	✓	✓		✓		
Graph Discovery	✓					
Nearest Neighbor Query	✓					✓
Resource Reservation			✓		✓	
Semantic-Awareness						✓
Proximity-Awareness					✓	

7. Conclusion

Resource discovery is one of the most significant challenging issues, particularly in large-scale distributed environments, which have already become mainstream platforms in academia and industry. It has a large variety of applications ranging from web-based resource discovery to job/task execution in high-performance computing clusters, while it can be implemented on top of different computing environments. Thus, due to the objectives and the purposes of designing a resource discovery approach, it can be seen through a wide range of concepts, methodologies, design and evaluation aspects. In this paper, we have discussed resource discovery solutions for large-scale distributed environments, with a particular focus on the design aspects. Accordingly, we have categorized all of these aspects in three classes: underlying aspects, design aspects, and evaluation aspects.

In underlying aspects, we have investigated and reviewed the current state of resource discovery protocols from the perspective of the structure. We highlighted that the overall characteristics of computing environments (e.g., Grid, Cloud, HPC, HTC, Cluster, etc.) have a large impact on most of the design and quality aspects (e.g., in terms of, objectives, methodology and even performance) of any potential proposal for resource discovery for those environments. Moreover, we presented the advantages and disadvantages of using different distributed architectures, such as centralized, hierarchical, decentralized and decentralized-

tree, as underlying information structures. Additionally, we have reviewed the literature from the aspects of ontology and resource description models and different clustering approaches.

In design aspects, we have covered most methods and techniques for resource discovery in several aspects, including discovery strategies, search methods, packet propagation techniques, information delivery, synchronization and query termination. However, we did not cover security and summarization aspects. This provides a clear understanding of the potential methods, their weaknesses, and strengths. Furthermore, we have reviewed the current works in state of the art considering those methods.

In evaluation aspects, we have introduced a set of most important issues for resource discovery in large-scale distributed computing systems. We have proposed that the major evaluation and performance factors for resource discovery assessment can be classified in five different aspects including scalability, efficiency, reliability, flexibility, and heterogeneity.

We expect that the future direction of research in resource discovery for distributed computing systems will expand upon the following key technology trends. These trends can already be felt incipiently in recent proposals.

-Highly scalable, decentralized resource discovery solutions are required for future large-scale systems. Current HPCs, Clusters, and Clouds mainly use centralized or hierarchical approaches for resource discovery. These approaches may provide

reasonable performance for current systems. However, sticking with such approaches for future HPCs, Clusters, and Clouds can provide big issues in terms of efficiency. Because in future large dimension systems it is not attainable to maintain and manage a huge amount of information in a central place. This would be more critical particularly for applications which need very detailed information of resources (e.g., details of processing cores and interconnects). On the other hand, dealing with dynamicity of resources and the environment in such large scale future systems using a centralized approach would be a challenging issue.

-Manycore technology would be in the core of the future computing. Accordingly, there would be a set of requirements for resource discovery in future manycore systems which include scalability, heterogeneity, and hierarchies. In fact, the scalability is required to cope with the large dimensions of cores, dies, and nodes in the system. The heterogeneity of cores, processors, nodes, networks and interconnects is also foreseeable in future systems. Therefore, resource discovery approaches must support heterogeneity of resources. Furthermore, hierarchies (like Node-Die-Core) are in the nature of future manycore systems. Thus, it would be essential for a discovery approach to be efficiently adapted to the hierarchical nature of such future systems. Referring to the high complexity of future manycore systems, we can mention further discovery requirements such as adaptability, querying flexibility, complex querying, query expressiveness, and resource graph discovery.

-Thread-level resource discovery is desired for future systems [297]. This can be defined as the capability of the discovery system to deal with the query conditions in thread-level (i.e., resource requirements for each thread in a process). In current literature, (P2P-based) decentralized discovery approaches are mostly used for Grids. However, these approaches generally work at task-level (due to the Grid nature) in which parallelism of independent tasks is exploited, resulting in a course-grained discovery. This makes current Grid discovery methods inadequate to deal with the manycore nature of future computing (in terms of efficient satisfaction of all query constraints) [349, 350], due to the lack of support for thread-level discovery as well as the high resolution of future manycore systems and future parallel applications. Nevertheless, Operating Systems can perform resource allocation in instruction-level. This provides motivation to go beyond the current Grids through the concept of Distributed Operation Systems (DOSs), capable to run on future large-scale systems. S[o]OS [295–297, 351–360], Barrelfish [361], and MyThOS [362] are recent research projects aiming to provide references for such DOSs. Resource discovery for DOSs requires a fine-grained, thread-level (system-level) resource discovery approach which can work in a fully decentralized, self-determining and autonomous fashion.

8. Acknowledgment

The authors acknowledge the support of project FP7-ICT-2009.8.1, Grant Agreement No.248465, Service-oriented Operating Systems (2010-2013) [295–297, 351–360] and of project

Cloud Thinking (2013-2015), CENTRO-07-ST24-FEDER-002031 [363].

References

- [1] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop, 2008. GCE'08, 2008, pp. 1–10.
- [2] G. Mateescu, W. Gentsch, C. J. Ribbens, Hybrid computing-where HPC meets grid and cloud computing, *Future Gener. Comput. Syst.* 27 (2011) 440–453.
- [3] N. J. Navimipour, A. M. Rahmani, A. H. Navin, M. Hosseinzadeh, Resource discovery mechanisms in grid systems: A survey, *Journal of Network and Computer Applications* 41 (2014) 389–410.
- [4] A. Hameurlain, D. Cokuslu, K. Erciyes, Resource discovery in grid systems; a survey, *Int. J. Metadata Semant. Ontologies* 5 (2010) 251–263.
- [5] Z. Xin-Xin, Z. Zhen-Wan, K. Peng, S. Ren-Jie, A survey of resource discovery in mobile peer-to-peer networks, in: *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, 2015, pp. 122–125.
- [6] M. I. Hassan, A. Abdullah, Semantic-based grid resource discovery systems a literature review and taxonomy, in: *2010 International Symposium on Information Technology*, volume 3, 2010, pp. 1286–1296.
- [7] F. Sharifkhani, M. R. Pakravan, A review of new advances in resource discovery approaches in unstructured P2P networks, in: *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, 2013, pp. 828–833.
- [8] A. Arunachalam, O. Sornil, An analysis of the overhead and energy consumption in flooding, random walk and gossip based resource discovery protocols in mp2p networks, in: *2015 Fifth International Conference on Advanced Computing Communication Technologies*, 2015, pp. 292–297.
- [9] D. Lazaro, J. M. Marques, J. Jorba, X. Vilajosana, Decentralized resource discovery mechanisms for distributed computing in peer-to-peer environments, *ACM Comput. Surv.* 45 (2013) 54:1–54:40.
- [10] E. Meshkova, J. Riihijärvi, M. Petrova, P. Mähönen, A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks, *Computer Networks* 52 (2008) 2097 – 2128.
- [11] B. S. Murugan, D. Lopez, A survey of resource discovery approaches in distributed computing environment, *International Journal of Computer Applications* 22 (2011) 44–46.
- [12] K. Vanthournout, G. Deconinck, R. Belmans, A taxonomy for resource discovery, *Personal Ubiquitous Comput.* 9 (2005) 81–89.
- [13] K. Vanthournout, G. Deconinck, R. Belmans, A taxonomy for resource discovery, in: C. Müller-Schloer, T. Ungerer, B. Bauer (Eds.), *Organic and Pervasive Computing – ARCS 2004: International Conference on Architecture of Computing Systems*, Augsburg, Germany, March 23–26, 2004. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 78–91.
- [14] S. Davtyan, Resource Discovery and Cooperation in Decentralized Systems, Ph.D. thesis, University of Connecticut, University of Connecticut, Storrs, Connecticut, United States, 2014. Doctoral Dissertations. Paper 425.
- [15] N. Antonopoulos, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications: Models, Methodologies and Applications*, IGI Global, 2010.
- [16] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, *Software: Practice and Experience* 32 (2002) 135–164.
- [17] D. Talia, P. Trunfio, Peer-to-peer protocols and grid services for resource discovery on grids, in: L. Grandinetti (Ed.), *Grid Computing The New Frontier of High Performance Computing*, volume 14 of *Advances in Parallel Computing*, North-Holland, 2005, pp. 83–103.
- [18] D. C. Erdil, Adaptive Dissemination Protocols for Hybrid Grid Resource Scheduling, Ph.D. thesis, State University of New York at Binghamton, Binghamton, NY, USA, 2007. AAI3289113.
- [19] A. K. Shaikh, S. M. Alhashmi, R. Parthiban, A semantic-based centralized resource discovery model for grid computing, in: *Grid and Distributed Computing*, Springer, 2011, pp. 161–170.

- [20] I. Foster, C. Kesselman, Globus: A metacomputing infrastructure toolkit, *International Journal of High Performance Computing Applications* 11 (1997) 115–128.
- [21] C. Germain, V. Neri, G. Fedak, F. Cappello, Xtremweb: building an experimental platform for global computing, in: *Grid Computing—GRID 2000*, Springer, 2000, pp. 91–101.
- [22] A. Chien, B. Calder, S. Elbert, K. Bhatia, Entropia: architecture and performance of an enterprise desktop grid system, *Journal of Parallel and Distributed Computing* 63 (2003) 597–610.
- [23] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, et al., Adaptive computing on the grid using apples, *Parallel and Distributed Systems, IEEE Transactions on* 14 (2003) 369–382.
- [24] A. K. Shaikh, S. M. Alhashmi, R. Parthiban, A semantic decentralized chord-based resource discovery model for grid computing, in: *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, 2011, pp. 142–148.
- [25] X. Wang, L. F. Kong, Resource clustering based decentralized resource discovery scheme in computing grid, in: *Machine Learning and Cybernetics, 2007 International Conference on*, volume 7, 2007, pp. 3859–3863.
- [26] R.-S. Chang, M.-S. Hu, A resource discovery tree using bitmap for grids, *Future Gener. Comput. Syst.* 26 (2010) 29–37.
- [27] L. M. Khanli, S. Kargar, Frdt: footprint resource discovery tree for grids, *Future Generation Computer Systems* 27 (2011) 148–156.
- [28] A. Forestiero, C. Mastroianni, G. Spezzano, A decentralized ant-inspired approach for resource management and discovery in grids, *Multiagent and Grid Systems* 3 (2007) 43–63.
- [29] Y. Yin, H. Cui, X. Chen, The grid resource discovery method based on hierarchical model, *Information Technology Journal* 6 (2007) 1090–1094.
- [30] H. Sun, J. Huai, Y. Liu, R. Buyya, Rct: A distributed tree for supporting efficient range and multi-attribute queries in grid computing, *Future Generation Computer Systems* 24 (2008) 631–643.
- [31] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid information services for distributed resource sharing, in: *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, 2001, pp. 181–194.
- [32] G. Alliance, Gt information services: Monitoring & discovery system (mds), 2005.
- [33] D. Abramson, J. Giddy, L. Kotler, High performance parametric modeling with nimrod/g: killer application for the global grid?, in: *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000, 2000*, pp. 520–528.
- [34] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-g: A computation management agent for multi-institutional grids, *Cluster Computing* 5 (2002) 237–246.
- [35] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, M. Craig, A. Djaoui, A. Duncan, S. Fisher, A. Gray, S. Hicks, S. Kenny, J. Leake, O. Lyttleton, J. Magowan, R. Middleton, W. Nutt, D. O’Callaghan, N. Podhorszki, P. Taylor, J. Walk, A. Wilson, Fault tolerance in the r-GMa information and monitoring system, in: *Proceedings of the 2005 European Conference on Advances in Grid Computing, EGC’05*, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 751–760.
- [36] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, A. Datta, A. Djaoui, L. Field, S. Fisher, S. Hicks, S. Kenny, J. Magowan, W. Nutt, D. O’Callaghan, M. Oevers, N. Podhorszki, J. Ryan, M. Soni, P. Taylor, A. Wilson, X. Zhu, The canonicalproducer: An instrument monitoring component of the relational grid monitoring architecture (r-GMa), in: *Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, ISPDC ’04*, IEEE Computer Society, Washington, DC, USA, 2004, pp. 232–237.
- [37] X. Zhang, J. L. Freschl, J. M. Schopf, Scalability analysis of three monitoring and information systems: Mds2, r-GMa, and hawkeye, *Journal of Parallel and Distributed Computing* 67 (2007) 883–902.
- [38] Z. Pan, A. Qasem, S. Kanitkar, F. Prabhakar, J. Heflin, Hawkeye: A practical large scale demonstration of semantic web integration, in: *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part II, OTM’07*, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 1115–1124.
- [39] J. Yu, S. Venugopal, R. Buyya, A market-oriented grid directory service for publication and discovery of grid service providers and their services, *J. Supercomput.* 36 (2006) 17–31.
- [40] J. Yu, S. Venugopal, R. Buyya, A market-oriented grid directory service for publication and discovery of grid service providers and their services, *J. Supercomput.* 36 (2006) 17–31.
- [41] X. Zhang, J. Schopf, Performance analysis of the globus toolkit monitoring and discovery service, mds2, in: *the 2004 Ieee International Performance, Computing, and Communications Conference, 2004*, pp. 843–849.
- [42] H. N. L. C. Keung, S. A. Jarvis, Performance modelling of a self-adaptive and self-optimising resource monitoring system for dynamic grid environments, 2004.
- [43] I. Diaz, G. Fernandez, M. J. Martinm, P. Gonzalez, J. Tourino, Integrating the common information model with mds4, in: *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing, GRID ’08*, IEEE Computer Society, Washington, DC, USA, 2008, pp. 298–303.
- [44] J. M. Schopf, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D’Arcy, A. Chervenak, Monitoring the grid with the globus toolkit MDS4, in: *Journal of Physics: Conference Series*, volume 46, IOP Publishing, IOP Publishing, 2006, p. 521.
- [45] M. L. Massie, B. N. Chun, D. E. Culler, The ganglia distributed monitoring system: design, implementation, and experience, *Parallel Computing* 30 (2004) 817 – 840.
- [46] H. Sun, J. Huai, Y. Liu, R. Buyya, RCT: A distributed tree for supporting efficient range and multi-attribute queries in grid computing, *Future Generation Computer Systems* 24 (2008) 631–643.
- [47] A. Naseer, L. K. Stergioulas, Resource discovery in grids and other distributed environments: States of the art, *Multiagent and Grid Systems* 2 (2006) 163–182.
- [48] K. Lee, T. Choi, P. O. Boykin, R. J. Figueiredo, Matchtree: Flexible, scalable, and fault-tolerant wide-area resource discovery with distributed matchmaking and aggregation, *Future Gener. Comput. Syst.* 29 (2013) 1596–1610.
- [49] G. Pipan, Use of the {TRIPOD} overlay network for resource discovery, *Future Generation Computer Systems* 26 (2010) 1257–1270.
- [50] T. Ghafarian, H. Deldari, B. Javadi, M. H. Yaghmaee, R. Buyya, Cycloid-grid: A proximity-aware P2P-based resource discovery architecture in volunteer computing systems, *Future Gener. Comput. Syst.* 29 (2013) 1583–1595.
- [51] H. Shen, A p2p-based intelligent resource discovery mechanism in internet-based distributed systems, *Journal of Parallel and Distributed Computing* 69 (2009) 197 – 209.
- [52] T. Welch, A technique for high-performance data compression, *Computer* 17 (1984) 8–19.
- [53] A. R. Butt, R. Zhang, Y. C. Hu, A self-organizing flock of condors, *Journal of Parallel and Distributed Computing* 66 (2006) 145 – 161.
- [54] R. Raman, M. Livny, M. Solomon, Matchmaking: Distributed resource management for high throughput computing, in: *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, 1998, pp. 140–146.
- [55] M. Litzkow, M. Livny, M. Mutka, Condor-a hunter of idle workstations, in: *Distributed Computing Systems, 1988., 8th International Conference on*, 1988, pp. 104–111.
- [56] T. Tannenbaum, D. Wright, K. Miller, M. Livny, Condor: A distributed job scheduler, in: *Beowulf Cluster Computing with Linux*, MIT Press, Cambridge, MA, USA, 2002, pp. 307–350.
- [57] M. Xu, S. Zhou, J. Guan, A new and effective hierarchical overlay structure for peer-to-peer networks, *Comput. Commun.* 34 (2011) 862–874.
- [58] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, A. Wolman, Skipnet: a scalable overlay network with practical locality properties, in: *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4, USITS’03*, USENIX Association, Berkeley, CA, USA, 2003, pp. 9–9.
- [59] J. Aspnes, G. Shah, Skip graphs, *ACM Trans. Algorithms* 3 (2007) 37.
- [60] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, *IEEE/ACM Trans. Netw.* 11 (2003) 17–32.
- [61] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, *SIGCOMM Comput. Commun. Rev.* 31 (2001) 161–172.

- [62] A. I. T. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Middleware '01, Springer-Verlag, London, UK, UK, 2001, pp. 329–350.
- [63] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. D. Kubiatowicz, Tapestry: a resilient global-scale overlay for service deployment, *IEEE J.Sel. A. Commun.* 22 (2006) 41–53.
- [64] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, R. Schmidt, P-grid: a self-organizing structured P2P system, *SIGMOD Rec.* 32 (2003) 29–33.
- [65] V. March, Y. M. Teo, X. Wang, Dgrid: a DHT-based resource indexing and discovery scheme for computational grids, in: Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68, ACSW '07, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2007, pp. 41–48.
- [66] A. R. Bharambe, M. Agrawal, S. Seshan, Mercury: Supporting scalable multi-attribute range queries, in: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '04, ACM, New York, NY, USA, 2004, pp. 353–366.
- [67] C. Tang, Z. Xu, M. Mahalingam, psearch: information retrieval in structured overlays, *SIGCOMM Comput. Commun. Rev.* 33 (2003) 89–94.
- [68] A. K. Shaikh, S. M. Alhashmi, R. Parthiban, A semantic decentralized chord-based resource discovery model for grid computing, in: Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems, ICPADS '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 142–148.
- [69] Z. Chen, L. Wu, J. Zhang, X. Hu, Y. Xu, Heuristic resource discovery in p2p network, in: Proceedings of the 25th international conference on Industrial Engineering and Other Applications of Applied Intelligent Systems: advanced research in applied artificial intelligence, IEA/AIE' 12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 333–342.
- [70] S. Sotiriadis, N. Bessis, N. Antonopoulos, Using self-led critical friend topology based on P2P chord algorithm for node localization within cloud communities, in: Proceedings of the 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 490–495.
- [71] A. Forestiero, C. Mastroianni, M. Meo, Self-chord: A bio-inspired algorithm for structured P2P systems, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 44–51.
- [72] X. Ke, S. Meina, S. Junde, An improved P2P lookup protocol model, *Cluster Computing* 13 (2010) 199–211.
- [73] S. Kniesburges, A. Koutsopoulos, C. Scheideler, Re-chord: a self-stabilizing chord overlay network, in: Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures, SPAA '11, ACM, New York, NY, USA, 2011, pp. 235–244.
- [74] E. J.-L. Lu, Y.-F. Huang, S.-C. Lu, ML-chord: A multi-layered P2P resource sharing model, *J. Netw. Comput. Appl.* 32 (2009) 578–588.
- [75] Y.-J. Joung, J.-C. Wang, Chord2: A two-layer chord for reducing maintenance overhead via heterogeneity, *Comput. Netw.* 51 (2007) 712–731.
- [76] B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph, Tapestry: An Infrastructure for Fault-tolerant Wide-area Location, Technical Report, University of California at Berkeley, Berkeley, CA, USA, 2001.
- [77] D. Batre, A. Hoing, M. Raack, U. Rerrer-Brusch, O. Kao, Extending pastry by an alphanumeric overlay, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 36–43.
- [78] Z. Guo, L. Min, S. Yang, H. Yang, An enhanced p4p-based pastry routing algorithm for P2P network, in: Proceedings of the 2010 IEEE International Conference on Granular Computing, GRC '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 687–691.
- [79] T. Wang, R.-H. Di, A semantic web service discovery model based on pastry system, in: Proceedings of the Fifth Annual ChinaGrid Conference, CHINAGRID '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 205–208.
- [80] J. Chen, L. Wang, Improved hierarchical network model based on pastry scheme, in: Proceedings of the 2009 Second International Workshop on Knowledge Discovery and Data Mining, WKDD '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 859–862.
- [81] D. Spence, T. Harris, Xenosearch: distributed resource discovery in the xenoserver open platform, in: High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on, 2003, pp. 216–225.
- [82] I. Chang-Yen, D. Smith, N.-F. Tzeng, Structured peer-to-peer resource discovery for computational grids, in: Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities, MG '08, ACM, New York, NY, USA, 2008, pp. 6:1–6:8.
- [83] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke, J. Shanmugasundaram, P-ring: an efficient and robust P2P range index structure, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, SIGMOD '07, ACM, New York, NY, USA, 2007, pp. 223–234.
- [84] A. N. Crainiceanu, Answering complex queries in peer-to-peer systems, Ph.D. thesis, Cornell University, Ithaca, NY, USA, 2006. AAI3227233.
- [85] P. Ganesan, M. Bawa, H. Garcia-Molina, Online balancing of range-partitioned data with applications to peer-to-peer systems, in: Proceedings of the Thirtieth international conference on Very large data bases - Volume 30, VLDB '04, VLDB Endowment, 2004, pp. 444–455.
- [86] G. Pirró, P. Trunfio, D. Talia, P. Missier, C. Goble, Ergot: A semantic-based system for service discovery in distributed infrastructures, in: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, 2010, pp. 263–272.
- [87] A. Crespo, H. Garcia-Molina, Semantic overlay networks for p2p systems, in: Proceedings of the Third international conference on Agents and Peer-to-Peer Computing, AP2PC'04, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 1–13.
- [88] Y. Tao, H. Jin, S. Wu, X. Shi, Scalable dht- and ontology-based information service for large-scale grids, *Future Generation Computer Systems* 26 (2010) 729 – 739.
- [89] E. Meshkova, J. Riihijärvi, M. Petrova, P. Mähönen, A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks, *Computer Networks* 52 (2008) 2097–2128.
- [90] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Pouyoul, B. Yeager, Project jxta 2.0 super-peer virtual network, 2003.
- [91] D. Talia, P. Trunfio, Peer-to-peer protocols and grid services for resource discovery on grids, *Advances in Parallel Computing* 14 (2005) 83–103.
- [92] K. Truelove, A. Chasin, Morpheus out of the underworld, *The O'Rielly Network* (2001).
- [93] A. Crespo, H. Garcia-Molina, Routing indices for peer-to-peer systems, in: *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, 2002, pp. 23–32.
- [94] M. Ripeanu, Peer-to-peer architecture case study: Gnutella network, in: *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, 2001, pp. 99–100.
- [95] S. Meng, C. Shi, D. Han, X. Zhu, Y. Yu, A statistical study of today's gnutella, in: *Proceedings of the 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development, APWeb'06*, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 189–200.
- [96] D. Talia, P. Trunfio, Adapting a pure decentralized peer-to-peer protocol for grid services invocation, *Parallel Processing Letters* 15 (2005) 67–84.
- [97] S. H. Kwok, K. Y. Chan, An enhanced gnutella p2p protocol: a search perspective, in: *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004., volume 1*, 2004, pp. 599–604.
- [98] A. C. Caminero, A. Robles-Gómez, S. Ros, R. Hernández, L. Tobarra, P2P-based resource discovery in dynamic grids allowing multi-attribute and range queries, *Parallel Computing* 39 (2013) 615–637.
- [99] R. Brunner, A. C. Caminero, O. F. Rana, F. Freitag, L. Navarro, Network-aware summarisation for resource discovery in P2P-content networks, *Future Generation Computer Systems* 28 (2012) 563–572.
- [100] J. Lee, P. Keleher, A. Sussman, Decentralized multi-attribute range search for resource discovery and load balancing, *The Journal of Supercomputing* 68 (2014) 890–913.

- [101] I. Foster, A. Iamnitchi, On death, taxes, and the convergence of peer-to-peer and grid computing, in: M. F. Kaashoek, I. Stoica (Eds.), *Peer-to-Peer Systems II: Second International Workshop, IPTPS 2003*, Berkeley, CA, USA, February 21–22, 2003. Revised Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 118–128.
- [102] D. Talia, P. Trunfio, Toward a synergy between P2P and grids, *Internet Computing*, IEEE 7 (2003) 96–95.
- [103] N. Hidalgo, L. Arantes, P. Sens, X. Bonnaire, Echo: Efficient complex query over {DHT} overlays, *Journal of Parallel and Distributed Computing* 88 (2016) 31–45.
- [104] J. Albrecht, et al., Design and implementation trade-offs for wide-area resource discovery, *Acm Transactions on Internet Technology* 8(4) (2008).
- [105] C. Mastroianni, D. Talia, O. Verta, A super-peer model for resource discovery services in large-scale grids, *Future Generation Computer Systems* 21 (2005) 1235–1248.
- [106] A. R. Butt, R. Zhang, Y. C. Hu, A self-organizing flock of condors, *Journal of Parallel and Distributed Computing* 66 (2006) 145–161.
- [107] J. A. Torkestani, A distributed resource discovery algorithm for {P2P} grids, *Journal of Network and Computer Applications* 35 (2012) 2028–2036.
- [108] S. Basu, S. Banerjee, P. Sharma, S.-J. Lee, Nodewiz: peer-to-peer resource discovery for grids, in: *Cluster Computing and the Grid*, 2005. CCGrid 2005. IEEE International Symposium on, volume 1, 2005, pp. 213–220Vol. 1.
- [109] S. Basu, L. Costa, F. Brasileiro, S. Banerjee, P. Sharma, S.-J. Lee, Nodewiz: Fault-tolerant grid information service, *Peer-to-Peer Networking and Applications* 2 (2009) 348–366.
- [110] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi, Peer-to-peer resource discovery in grids: Models and systems, *Future Generation Computer Systems* 23 (2007) 864–878.
- [111] I. Foster, C. Kesselman, S. Tuecke, Chapter 17 - The Open Grid Services Architecture, *The Morgan Kaufmann Series in Computer Architecture and Design*, Morgan Kaufmann, Burlington, 2 edition, 2004.
- [112] M. Harchol-Balter, T. Leighton, D. Lewin, Resource discovery in distributed networks, in: *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, PODC '99*, ACM, New York, NY, USA, 1999, pp. 229–237.
- [113] D. A. Reed, Grids, the teragrid, and beyond, *Computer* 36 (2003) 62–68.
- [114] E. Imamagić, B. Radić, D. Dobrenić, Job management systems analysis, in: *6th CARNET Users' Conference*, Zagreb, Croatia, 2004, 2004, pp. 1–9.
- [115] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-g: A computation management agent for multi-institutional grids, *Cluster Computing* 5 (2002) 237–246.
- [116] G. Sabin, V. Sahasrabudhe, P. Sadayappan, Assessment and enhancement of meta-schedulers for multi-site job sharing, in: *High Performance Distributed Computing*, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium on, 2005, pp. 144–153.
- [117] D. Anderson, Boinc: a system for public-resource computing and storage, in: *Grid Computing*, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, 2004, pp. 4–10.
- [118] I. Foster, Globus toolkit version 4: Software for service-oriented systems, in: *Proceedings of the 2005 IFIP International Conference on Network and Parallel Computing, NPC'05*, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 2–13.
- [119] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: A fast and light-weight task execution framework, in: *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC '07*, ACM, New York, NY, USA, 2007, pp. 43:1–43:12.
- [120] G. Staples, Torque resource manager, in: *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC '06*, ACM, New York, NY, USA, 2006, Article No.8.
- [121] S. Koussih, A. Acharya, S. Setia, Dodo: A user-level system for exploiting idle memory in workstation clusters, in: *High Performance Distributed Computing*, 1999. Proceedings. The Eighth International Symposium on, 1999, pp. 301–308.
- [122] S. Koussih, A. Acharya, S. Setia, Dodo: a user-level system for exploiting idle memory in workstation clusters, in: *High Performance Distributed Computing*, 1999. Proceedings. The Eighth International Symposium on, 1999, pp. 301–308.
- [123] A. B. Yoo, M. A. Jette, M. Grondona, Slurm: Simple linux utility for resource management, in: *Job Scheduling Strategies for Parallel Processing*, 2003, pp. 44–60.
- [124] R. L. Henderson, Job scheduling under the portable batch system, in: *Job scheduling strategies for parallel processing*, 1995, pp. 279–294.
- [125] A. Chien, B. Calder, S. Elbert, K. Bhatia, Entropia: Architecture and performance of an enterprise desktop grid system, *J. Parallel Distrib. Comput.* 63 (2003) 597–610.
- [126] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, Seti@home: An experiment in public-resource computing, *Commun. ACM* 45 (2002) 56–61.
- [127] V. S. Pande, I. Baker, J. Chapman, S. P. Elmer, S. Khaliq, S. M. Larson, Y. M. Rhee, M. R. Shirts, C. D. Snow, E. J. Sorin, et al., Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing, *Biopolymers* 68 (2003) 91–109.
- [128] S. Zhou, LSf: Load sharing in large heterogeneous distributed systems, in: *I Workshop on Cluster Computing*, 1992.
- [129] S. Kannan, M. Roberts, P. Mayes, D. Brelford, J. F. Skovira, Workload management with loadleveler, *IBM Redbooks* 2 (2001) 2.
- [130] A. Keller, A. Reinefeld, Ccs resource management in networked hpc systems, in: *Proceedings of the Seventh Heterogeneous Computing Workshop, HCW '98*, IEEE Computer Society, Washington, DC, USA, 1998, pp. 44–.
- [131] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Gener. Comput. Syst.* 25 (2009) 599–616.
- [132] S. Di, C.-L. Wang, L. Chen, Ex-post efficient resource allocation for self-organizing cloud, *Comput. Electr. Eng.* 39 (2013) 2342–2356.
- [133] D. C. Erdil, Dependable autonomic cloud computing with information proxies, in: *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, 2011 IEEE International Symposium on, 2011, pp. 1518–1524.
- [134] W. Gentsch, Porting hpc applications to grids and clouds, *Cloud, Grid and High Performance Computing: Emerging Applications* (2011) 10–38.
- [135] F. Eljorje, J. Lee, Cloudswitch: A state-aware monitoring strategy towards energy-efficient and performance-aware cloud data centers., *KSII Transactions on Internet & Information Systems* 9 (2015).
- [136] B. Di Martino, G. Cretella, A. Esposito, Cross-platform cloud apis, in: *Cloud Portability and Interoperability*, Springer, 2015, pp. 45–57.
- [137] A. Brogi, A. Ibrahim, J. Soldani, J. Carrasco, J. Cubo, E. Pimentel, F. D'Andria, Seaclouds: a european project on seamless management of multi-cloud applications, *ACM SIGSOFT Software Engineering Notes* 39 (2014) 1–4.
- [138] A. Ionescu, Standard interfaces for open source infrastructure as a service platforms, *Informatica Economica* 19 (2015) 68.
- [139] D. Nurmi, R. Wolski, C. Grzegorzczak, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The eucalyptus open-source cloud-computing system, in: *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09*, IEEE Computer Society, Washington, DC, USA, 2009, pp. 124–131.
- [140] Y. Chen, X. Li, F. Chen, Overview and analysis of cloud computing research and application, in: *E-Business and E-Government (ICEE)*, 2011 International Conference on, 2011, pp. 1–4.
- [141] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, Virtual infrastructure management in private and hybrid clouds, *Internet computing*, IEEE 13 (2009) 14–22.
- [142] S. S. Manvi, G. K. Shyam, Resource management for infrastructure as a service (iaas) in cloud computing: A survey, *Journal of Network and Computer Applications* 41 (2014) 424–440.
- [143] D. Milojicic, I. M. Llorente, R. S. Montero, Opennebula: A cloud management tool, *IEEE Internet Computing* 15 (2011) 11–14.
- [144] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emerich, F. Galan, The reservoir model and architecture for open federated cloud computing, *IBM Journal of Research and Development* 53 (2009) 4:1–4:11.
- [145] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muñoz, G. Tofetti, Reservoir - when one cloud is not enough, *Computer* 44 (2011) 44–51.

- [146] P. Wright, Y. L. Sun, T. Harmer, A. Keenan, A. Stewart, R. Perrott, A constraints-based resource discovery model for multi-provider cloud environments, *Journal of Cloud Computing* 1 (2012) 1–14.
- [147] M. Siddiqui, T. Fahringer, Semantics in the grid: Towards ontology-based resource provisioning, in: *Grid Resource Management: On-demand Provisioning, Advance Reservation, and Capacity Planning of Grid Resources*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 157–177.
- [148] E. Newcomer, *Understanding Web Services: XML, WSDL, SOAP, and UDDI*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [149] C. Campo, M. Munoz, J. C. Perea, A. Marin, C. Garcia-Rubio, PDP and gsd: A new service discovery middleware to support spontaneous interactions in pervasive systems, in: *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOMW '05*, IEEE Computer Society, Washington, DC, USA, 2005, pp. 178–182.
- [150] S. Banerjee, S. Basu, S. Garg, S. Garg, S.-J. Lee, P. Mullan, P. Sharma, Scalable grid service discovery based on uddi, in: *Proceedings of the 3rd International Workshop on Middleware for Grid Computing, MGC '05*, ACM, New York, NY, USA, 2005, pp. 1–6.
- [151] A. Zhu, Y. Xie, Publishing and discovering knowledge services for design on uddi, *Int. J. Comput. Appl. Technol.* 23 (2005) 31–40.
- [152] S. Pastore, The service discovery methods issue: A web services uddi specification framework integrated in a grid environment, *J. Netw. Comput. Appl.* 31 (2008) 93–107.
- [153] K. Tutschku, V. A. Mehri, A. Carlsson, K. V. Chivukula, J. Christenson, On resource description capabilities of on-board tools for resource management in cloud networking and NFv infrastructures, in: *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016, pp. 442–447.
- [154] V. Vaikuntanathan, P. Voulgaris, Attribute based encryption using lattices, 2016. US Patent 20,160,156,465.
- [155] S. Fugkeaw, H. Sato, Design and implementation of collaborative ciphertext-policy attribute-role based encryption for data access control in cloud, *Journal of Information Security Research* 6 (2015) 71–84.
- [156] G. Baranwal, D. P. Vidyarthi, A fair multi-attribute combinatorial double auction model for resource allocation in cloud computing, *Journal of Systems and Software* 108 (2015) 60–76.
- [157] X. Yao, Z. Chen, Y. Tian, A lightweight attribute-based encryption scheme for the internet of things, *Future Generation Computer Systems* 49 (2015) 104–112.
- [158] N. S. Kumar, G. R. Lakshmi, B. Balamurugan, Enhanced attribute based encryption for cloud computing, *Procedia Computer Science* 46 (2015) 689–696.
- [159] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, W. Shi, Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts, *Information Sciences* 275 (2014) 370–384.
- [160] I. Sander, A. Jantsch, System modeling and transformational design refinement in forsyde [formal system design], *Trans. Comp.-Aided Des. Integ. Cir. Sys.* 23 (2006) 17–32.
- [161] A. Gill, T. Bull, A. Farmer, G. Kimmell, E. Komp, Types and type families for hardware simulation and synthesis: The internals and externals of kansas lava, in: *Proceedings of the 11th International Conference on Trends in Functional Programming, TFP'10*, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 118–133.
- [162] C. Baaij, M. Kooijman, J. Kuper, A. Boeijink, M. Gerards, Clash: Structural descriptions of synchronous hardware using haskell, in: *Digital System Design: Architectures, Methods and Tools (DSD)*, 2010 13th Euromicro Conference on, 2010, pp. 714–721.
- [163] J. Kuper, C. Baaij, M. Kooijman, M. Gerards, Exercises in architecture specification using c #x03Bb:ash, in: *Specification Design Languages (FDL 2010)*, 2010 Forum on, 2010, pp. 1–6.
- [164] G. Klyne, J. J. Carroll, B. McBride, Resource description framework (rdf): Concepts and abstract syntax, *World Wide Web Consortium*, <https://www.w3.org/TR/rdf11-concepts/> 10 (2014).
- [165] M. P. Ebenhoch, Legal knowledge representation using the resource description framework (rdf), in: *12th International Workshop on Database and Expert Systems Applications*, 2001, pp. 369–373.
- [166] K. S. Candan, H. Liu, R. Suvama, Resource description framework: Metadata and its applications, *SIGKDD Explor. Newsl.* 3 (2001) 6–19.
- [167] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, M. C. A. Klein, Oil in a nutshell, in: *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW '00*, Springer-Verlag, London, UK, UK, 2000, pp. 1–16.
- [168] I. Horrocks, Daml+oil: A reason-able web ontology language, in: *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '02*, Springer-Verlag, London, UK, UK, 2002, pp. 2–13.
- [169] D. L. McGuinness, R. Fikes, J. Hendler, L. A. Stein, Daml+oil: An ontology language for the semantic web, *IEEE Intelligent Systems* 17 (2002) 72–80.
- [170] L. Bangyong, T. Jie, L. Juanzi, W. Kehong, Using daml+oil to enhance search semantic, in: *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, WI '04*, IEEE Computer Society, Washington, DC, USA, 2004, pp. 465–468.
- [171] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, Reviewing the design of daml+oil: An ontology language for the semantic web, in: *Eighteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, 2002, pp. 792–797.
- [172] M. H. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. V. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, K. P. Sycara, Daml-s: Web service description for the semantic web, in: *Proceedings of the First International Semantic Web Conference on The Semantic Web, ISWC '02*, Springer-Verlag, London, UK, UK, 2002, pp. 348–363.
- [173] H. Ballani, P. Francis, Towards a global ip anycast service, *SIGCOMM Comput. Commun. Rev.* 35 (2005) 301–312.
- [174] T. Stevens, M. De Leenheer, C. Develder, F. De Turck, B. Dhoedt, P. De-meester, Astas: Architecture for scalable and transparent anycast services, *Communications and Networks, Journal of* 9 (2007) 457–465.
- [175] A. Montresor, A robust protocol for building superpeer overlay topologies, in: *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, 2004, pp. 202–209.
- [176] B. Beverly Yang, H. Garcia-Molina, Designing a super-peer network, in: *Data Engineering, 2003. Proceedings. 19th International Conference on*, 2003, pp. 49–60.
- [177] M. Montebello, C. Abela, Daml enabled web services and agents in the semantic web, in: *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, Springer-Verlag, London, UK, UK, 2003, pp. 46–58.
- [178] S. A. Ludwig, S. M. S. Reyhani, Introduction of semantic matchmaking to grid computing, *Journal of Parallel and Distributed Computing*, 65 (2005) 1533–1541.
- [179] C. Zhou, L.-T. Chia, B.-S. Lee, Service discovery and measurement based on daml-QoS ontology, in: *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, WWW '05*, ACM, New York, NY, USA, 2005, pp. 1070–1071.
- [180] D. K. Bimson, D. R. Hull, D. Nieten, The lexical bridge: A methodology for bridging the semantic gaps between a natural language and an ontology, in: *Semantic Web: Implications for Technologies and Business Practices*, Springer International Publishing, 2016, pp. 137–151.
- [181] B. D. Martino, G. Cretella, A. Esposito, A. Willner, A. Alloush, D. Bernstein, D. Vij, J. Weinman, Towards an ontology-based intercloud resource catalogue – the IEEE p2302 intercloud approach for a semantic resource exchange, in: *Proceedings of the 2015 IEEE International Conference on Cloud Engineering, IC2E '15*, IEEE Computer Society, Washington, DC, USA, 2015, pp. 458–464.
- [182] E. Meshkova, J. Riihijärvi, M. Petrova, P. Mähönen, A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks, *Comput. Netw.* 52 (2008) 2097–2128.
- [183] G. P. Jesi, A. Montresor, O. Babaoglu, Proximity-aware superpeer overlay topologies, in: *Proceedings of the Second IEEE International Conference on Self-Managed Networks, Systems, and Services, SelfMan'06*, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 43–57.
- [184] M. Raack, D. Battré, A. Höing, O. Kao, Papnet: A proximity-aware alphanumeric overlay supporting ganesan on-line load balancing, in: *Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems, ICPADS '09*, IEEE Computer Society, Washington, DC, USA, 2009, pp. 440–447.
- [185] Y.-J. Joung, Approaching neighbor proximity and load balance for range query in P2P networks, *Comput. Netw.* 52 (2008) 1451–1472.
- [186] Y.-J. Joung, W.-T. Wong, H.-M. Huang, Y.-F. Chou, Building a network-

- aware and load-balanced peer-to-peer system for range queries, *Comput. Netw.* 56 (2012) 2148–2167.
- [187] H. Jin, F. Luo, Q. Zhang, X. Liao, H. Zhang, *Gtapestry: A locality-aware overlay network for high performance computing*, in: *Proceedings of the 11th IEEE Symposium on Computers and Communications, ISCC '06*, IEEE Computer Society, Washington, DC, USA, 2006, pp. 76–81.
- [188] Y. Sun, L. Sun, X. Huang, Y. Lin, *Resource discovery in locality-aware group-based semantic overlay of peer-to-peer networks*, in: *Proceedings of the 1st International Conference on Scalable Information Systems, InfoScale '06*, ACM, New York, NY, USA, 2006, Article No.40.
- [189] J. Bo, *Heterogeneity-aware group-based semantic overlay network for P2P systems*, in: *Proceedings of the 2009 International Conference on Web Information Systems and Mining, WISM '09*, IEEE Computer Society, Washington, DC, USA, 2009, pp. 701–704.
- [190] X. Sun, K. Li, Y. Liu, Y. Tian, *Slup: A semantic-based and location-aware unstructured P2P network*, in: *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, HPC '08*, IEEE Computer Society, Washington, DC, USA, 2008, pp. 288–295.
- [191] H. Ying, *Clsp2p: A P2P overlay combination location and semantic clustering*, in: *Proceedings of the 2010 Third International Symposium on Information Processing, ISIP '10*, IEEE Computer Society, Washington, DC, USA, 2010, pp. 143–147.
- [192] J. Li, *Grid resource discovery based on semantically linked virtual organizations*, *Future Gener. Comput. Syst.* 26 (2010) 361–373.
- [193] C. Xiao, L. Nianzu, *Overlay construction within diffserv domains for QoS-aware multicasting*, in: *Proceedings of the 2010 Third International Symposium on Information Science and Engineering, ISISE '10*, IEEE Computer Society, Washington, DC, USA, 2010, pp. 271–274.
- [194] W. Li, Y. Wang, C. Li, S. Lu, D. Chen, *A QoS-aware service selection algorithm for multimedia service overlay networks*, in: *Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 01, ICPADS '07*, IEEE Computer Society, Washington, DC, USA, 2007, pp. 1–8.
- [195] E. P. Duarte, L. Z. Granville, L. Pirmez, J. N. Souza, R. C. Andrade, L. Tarouco, R. B. Correia, A. Lages, *Gigamanp2p: An overlay network for distributed QoS management and resilient routing*, *Int. J. Netw. Manag.* 22 (2011) 50–64.
- [196] J. Li, *Grid resource discovery based on semantically linked virtual organizations*, *Future Generation Computer Systems* 26 (2010) 361–373.
- [197] M. Li, D. B. Hoang, *Fiac: A resource discovery-based two-level admission control for differentiated service networks*, *Comput. Commun.* 28 (2005) 2094–2104.
- [198] R. Rosen, *Resource management: Linux kernel namespaces and cgroups*, *Haifux*, May 186 (2013).
- [199] W.-C. Chung, C.-J. Hsu, K.-C. Lai, K.-C. Li, Y.-C. Chung, *Direction-aware resource discovery in large-scale distributed computing environments*, *J. Supercomput.* 66 (2013) 229–248.
- [200] D. I. Wolinsky, Y. Liu, P. S. Juste, G. Venkatasubramanian, R. Figueiredo, *On the design of scalable, self-configuring virtual networks*, in: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09*, ACM, New York, NY, USA, 2009, pp. 13:1–13:12.
- [201] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, R. Boutaba, *Resource and service discovery in large-scale multi-domain networks*, *Communications Surveys Tutorials*, IEEE 9 (2007) 2–30.
- [202] J. Kempf, J. Goldschmidt, *Notification and subscription for slp*, 2001. RFC 3082.
- [203] Z. Xu, L. Chen, G. Gu, C. Kruegel, *Peerpress: Utilizing enemies' P2P strength against them*, in: *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12*, ACM, New York, NY, USA, 2012, pp. 581–592.
- [204] A. Brocco, A. Malatras, B. Hirsbrunner, *Proactive information caching for efficient resource discovery in a self-structured grid*, in: *Proceedings of the 2009 Workshop on Bio-Inspired Algorithms for Distributed Systems, BADS '09*, ACM, New York, NY, USA, 2009, pp. 11–18.
- [205] V. Iyengar, S. Tilak, M. J. Lewis, N. B. Abu-Ghazaleh, *Non-uniform information dissemination for dynamic grid resource discovery*, in: *Network Computing and Applications, 2004. (NCA 2004)*, Proceedings. Third IEEE International Symposium on, 2004, pp. 97–106.
- [206] S. Di, C.-L. Wang, *Decentralized proactive resource allocation for maximizing throughput of p2p grid*, *J. Parallel Distrib. Comput.* 72 (2012) 308–321.
- [207] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.
- [208] H. Barjini, M. Othman, H. Ibrahim, N. I. Udzir, *Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured p2p networks*, *Peer-to-Peer Networking and Applications* 5 (2012) 1–13.
- [209] V. V. Dimakopoulos, E. Pitoura, *On the performance of flooding-based resource discovery*, *Parallel and Distributed Systems, IEEE Transactions on* 17 (2006) 1242–1252.
- [210] R. Gaeta, M. Sereno, *On the evaluation of flooding-based search strategies in peer-to-peer networks*, *Concurrency and Computation: Practice and Experience* 20 (2008) 713–734.
- [211] B. Awerbuch, R. G. Gallager, *A new distributed algorithm to find breadth first search trees*, *IEEE Trans. Inf. Theor.* 33 (1987) 315–322.
- [212] R. Zhou, E. A. Hansen, *Breadth-first heuristic search*, *Artif. Intell.* 170 (2006) 385–408.
- [213] H. Shang, M. Kitsuregawa, *Efficient breadth-first search on large graphs with skewed degree distributions*, in: *Proceedings of the 16th International Conference on Extending Database Technology, EDBT '13*, ACM, New York, NY, USA, 2013, pp. 311–322.
- [214] S. A. M. Makki, G. Havas, *Distributed algorithms for depth-first search*, *Inf. Process. Lett.* 60 (1996) 7–12.
- [215] S. Makki, G. Havas, *Distributed algorithms for constructing a depth-first-search tree*, in: *Parallel Processing, 1994. Vol. 1. ICPP 1994. International Conference on*, volume 3, 1994, pp. 270–273.
- [216] C. Rhee, Y. D. Liang, S. K. Dhall, S. Lakshminarayanan, *Efficient algorithms for finding depth-first and breadth-first search trees in permutation graphs*, *Inf. Process. Lett.* 49 (1994) 45–50.
- [217] R. E. Korf, *Depth-limited search for real-time problem solving*, *Real-Time Systems* 2 (1990) 7–24.
- [218] R. E. Korf, *Iterative-deepening-a: an optimal admissible tree search*, in: *Proceedings of the 9th international joint conference on Artificial intelligence-Volume 2, 1985*, pp. 1034–1036.
- [219] A. Felner, *Position paper: Dijkstra's algorithm versus uniform cost search or a case against dijkstra's algorithm*, in: *Fourth Annual Symposium on Combinatorial Search, 2011*, pp. 47–51.
- [220] N. Bisnik, A. Abouzeid, *Modeling and analysis of random walk search algorithms in p2p networks*, in: *Hot topics in peer-to-peer systems, 2005. HOT-P2P 2005. Second International Workshop on*, 2005, pp. 95–103.
- [221] N. Shenvi, J. Kempe, K. B. Whaley, *Quantum random-walk search algorithm*, *Physical Review A* 67 (2003) 052307.
- [222] N. Bisnik, A. A. Abouzeid, *Optimizing random walk search algorithms in p2p networks*, *Computer Networks* 51 (2007) 1499–1514.
- [223] M. Jelasiy, M. van Steen, *Large-Scale Newscast Computing on the Internet*, Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, 2002.
- [224] M. Zaharia, S. Keshav, *Gossip-based search selection in hybrid peer-to-peer networks*, *Concurrency and Computation: Practice and Experience* 20 (2008) 139–153.
- [225] H. Chen, H. Jin, Y. Liu, L. M. Ni, *Difficulty-aware hybrid search in peer-to-peer networks*, *Parallel and Distributed Systems, IEEE Transactions on* 20 (2009) 71–82.
- [226] K. Oikonomou, D. Kogias, I. Stavrakakis, *Probabilistic flooding for efficient information dissemination in random graph topologies*, *Comput. Netw.* 54 (2010) 1615–1629.
- [227] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky, *Bimodal multicast*, *ACM Trans. Comput. Syst.* 17 (1999) 41–88.
- [228] U. Datta, A. Mukherjee, P. Sahu, S. Kundu, *Resource utilization of multi-hop CDMA wireless sensor networks with efficient forwarding protocols*, *Procedia Engineering* 64 (2013) 46–55.
- [229] S. Ferretti, *Gossiping for resource discovering: An analysis based on complex network theory*, *Future Gener. Comput. Syst.* 29 (2013) 1631–1644.
- [230] A. Ganesh, A.-M. Kermarrec, L. Massoulie, *Peer-to-peer membership management for gossip-based protocols*, *Computers, IEEE Transactions on* 52 (2003) 139–149.
- [231] S. Tang, E. Jaho, I. Stavrakakis, I. Koukoutsidis, P. V. Miegheem, *Modeling gossip-based content dissemination and search in distributed networking*, *Comput. Commun.* 34 (2011) 765–779.

- [232] M. Jelasity, A. Montresor, O. Babaoglu, T-man: Gossip-based fast overlay topology construction, *Comput. Netw.* 53 (2009) 2321–2339.
- [233] V. Kalogeraki, D. Gunopulos, D. Zenalipour-Yazti, A local search mechanism for peer-to-peer networks, in: Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM '02, ACM, New York, NY, USA, 2002, pp. 300–307.
- [234] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: Proceedings of the 16th International Conference on Supercomputing, ICS '02, ACM, New York, NY, USA, 2002, pp. 84–95.
- [235] Y. Zhang, L. Liu, Distance-aware bloom filters: Enabling collaborative search for efficient resource discovery, *Future Gener. Comput. Syst.* 29 (2013) 1621–1630.
- [236] S. Rhea, J. Kubiatowicz, Probabilistic location and routing, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, 2002, pp. 1248–1257 vol.3.
- [237] A. Crespo, H. Garcia-Molina, Routing indices for peer-to-peer systems, in: Distributed Computing Systems, Proceedings. 22nd International Conference on, 2002, pp. 23–32.
- [238] C. Yang, J. Wu, Dominating-set-based searching in peer-to-peer networks, in: M. Li, X.-H. Sun, Q. ni Deng, J. Ni (Eds.), *Grid and Cooperative Computing*, volume 3032 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2004, pp. 332–339.
- [239] S. Kutten, D. Peleg, U. Vishkin, Deterministic resource discovery in distributed networks, in: Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '01, ACM, New York, NY, USA, 2001, pp. 77–83.
- [240] B. Awerbuch, Y. Shiloach, New connectivity and msf algorithms for shuffle-exchange network and pram, *IEEE Trans. Comput.* 36 (1987) 1258–1263.
- [241] I. Cidon, I. Gopal, S. Kutten, New models and algorithms for future networks, in: Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC '88, ACM, New York, NY, USA, 1988, pp. 79–89.
- [242] P. Chaudhuri, An $O(\log n)$ parallel algorithm for strong connectivity augmentation problem, *International Journal of Computer Mathematics* 22 (1987) 187–197.
- [243] I. Abraham, D. Dolev, Asynchronous resource discovery, in: Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing, PODC '03, ACM, New York, NY, USA, 2003, pp. 143–150.
- [244] S. Kutten, D. Peleg, Asynchronous resource discovery in peer-to-peer networks, *Comput. Netw.* 51 (2007) 190–206.
- [245] M. Pathan, R. Buyya, Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery networks, *J. Netw. Comput. Appl.* 32 (2009) 976–990.
- [246] T. Alam, Z. Raza, An adaptive threshold based hybrid load balancing scheme with sender and receiver initiated approach using random information exchange, *Concurrency and Computation: Practice and Experience* 28 (2016) 2729–2746.
- [247] L. Watkins, R. Beyah, C. Corbett, Using network traffic to passively detect under utilized resources in high performance cluster grid computing environments, in: Proceedings of the First International Conference on Networks for Grid Applications, GridNets '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2007, pp. 16:1–16:8.
- [248] C. Law, K.-Y. Siu, An $O(\log n)$ randomized resource discovery algorithm, in: Brief Announcements of the 14th International Symposium on Distributed Computing, Technical University of Madrid, Technical Report FIM/110.1/DLSIIS, 2000, pp. 5–8.
- [249] S. Kutten, D. Peleg, Deterministic distributed resource discovery, in: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, 2000, p. 336.
- [250] V. K. Garg, A. Aziz, An Efficient Deterministic Algorithm for the Resource Discovery Problem, Technical Report, Technical Report, ENS 527, The University of Texas, Austin TX 78712, 2000.
- [251] D. Tate, G. Steven, F. Steven, Static scheduling for out-of-order instruction issue processors, in: Computer Architecture Conference, 2000. ACAC 2000. 5th Australasian, 2000, pp. 90–96.
- [252] A.-M. Kermarrec, M. van Steen, Gossiping in distributed systems, *SIGOPS Oper. Syst. Rev.* 41 (2007) 2–7.
- [253] G. D'Angelo, S. Ferretti, M. Marzolla, Adaptive event dissemination for peer-to-peer multiplayer online games, in: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, SIMU-Tools '11, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2011, pp. 312–319.
- [254] C. Georgiou, S. Gilbert, D. R. Kowalski, Meeting the deadline: On the complexity of fault-tolerant continuous gossip, in: Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '10, ACM, New York, NY, USA, 2010, pp. 247–256.
- [255] P. Costa, M. Migliavacca, G. P. Picco, G. Cugola, Introducing reliability in content-based publish-subscribe through epidemic algorithms, in: Proceedings of the 2nd International Workshop on Distributed Event-based Systems, DEBS '03, ACM, New York, NY, USA, 2003, pp. 1–8.
- [256] P. M. Melliar-Smith, L. E. Moser, I. Michel Lombera, Y.-T. Chuang, itrust: Trustworthy information publication, search and retrieval, in: Proceedings of the 13th International Conference on Distributed Computing and Networking, ICDCN '12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 351–366.
- [257] E. Simonton, B. Choi, S. Seidel, Using gossip for dynamic resource discovery, in: Parallel Processing, 2006. ICPP 2006. International Conference on, 2006, pp. 319–328.
- [258] A. Brocco, A. Malatras, B. Hirsbrunner, Enabling efficient information discovery in a self-structured grid, *Future Gener. Comput. Syst.* 26 (2010) 838–846.
- [259] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, *Inf. Sci.* 192 (2012) 120–142.
- [260] S. Ghosh, I. W. Marshall, Simple model of collective decision making during nectar source selection by honey bees, in: Workshop on Memory and Learning Mechanisms in Autonomous Robotics as part of the 8th European Conference on Artificial Life (ECAL 2005), 2005.
- [261] V. Tereshko, Reaction-diffusion model of a honeybee colony's foraging behaviour, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2000, pp. 807–816.
- [262] S. Y. Ko, I. Gupta, Y. Jo, A new class of nature-inspired algorithms for self-adaptive peer-to-peer computing, *ACM Trans. Auton. Adapt. Syst.* 3 (2008) 11:1–11:34.
- [263] J. Taheri, Y. Choon Lee, A. Y. Zomaya, H. J. Siegel, A bee colony based optimization approach for simultaneous job scheduling and data replication in grid environments, *Comput. Oper. Res.* 40 (2013) 1564–1578.
- [264] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, K. V. Price (Eds.), *New Ideas in Optimization*, McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999, pp. 11–32.
- [265] M. Dorigo, V. Maniezzo, A. Colomi, M. Dorigo, M. Dorigo, V. Maniezzo, V. Maniezzo, A. Colomi, A. Colomi, Positive Feedback as a Search Strategy, Technical Report, No. 91-016, Politecnico di Milano, Italy, 1991.
- [266] K. Krynicki, J. Jaen, J. A. Mocholi, On the performance of ACo-based methods in P2P resource discovery, *Appl. Soft Comput.* 13 (2013) 4813–4831.
- [267] T. Abdullah, A. Anjum, N. Bessis, S. Sotiriadis, K. Bertels, Nature inspired self organization for adhoc grids, in: Proceedings of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications, AINA '13, IEEE Computer Society, Washington, DC, USA, 2013, pp. 682–689.
- [268] Y. Deng, F. Wang, A. Ciura, Ant colony optimization inspired resource discovery in P2P grid systems, *J. Supercomput.* 49 (2009) 4–21.
- [269] E. Michlmayr, Ant algorithms for search in unstructured peer-to-peer networks, in: Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDEW '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 142–147.
- [270] G. C. Valdez, A self-adaptive ant colony system for semantic query routing problem in p2p networks, *Computación y Sistemas* 13 (2010) 433–448.
- [271] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *Evolutionary Computation, IEEE Transactions on* 1 (1997) 53–66.

- [272] T. Stützle, H. Hoos, Improvements on the ant-system: Introducing the max-min ant system, in: *Artificial Neural Nets and Genetic Algorithms*, Springer Vienna, 1998, pp. 245–249.
- [273] H. Yousefpour, Z. Jafari, Using neural search approach for resource discovery in P2P networks, *Procedia Computer Science* 3 (2011) 1512–1516.
- [274] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.
- [275] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, J. Vuori, Resource discovery in P2P networks using evolutionary neural networks, in: *International Conference on Advances in Intelligent Systems-Theory and Applications (AISTA 2004)*, 2004.
- [276] N. Ganguly, G. Canright, A. Deutsch, Design of an efficient search algorithm for P2P networks using concepts from natural immune systems, in: X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tiño, A. Kabán, H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2004, pp. 491–500.
- [277] S. Brown, Microsoft’s viral search displays your content in real time as it spreads across twitter, *Science* (2014).
- [278] S. Goel, A. Anderson, J. Hofman, D. J. Watts, The structural virality of online diffusion, *Management Science* 62 (2016) 180–196.
- [279] V. V. Dimakopoulos, E. Pitoura, A peer-to-peer approach to resource discovery in multi-agent systems, in: *Cooperative Information Agents VII*, Springer, 2003, pp. 62–77.
- [280] F. W. Glover, M. Laguna, *Tabu Search*, Springer, 1997.
- [281] S. C. Yusta, Different metaheuristic strategies to solve the feature selection problem, *Pattern Recogn. Lett.* 30 (2009) 525–534.
- [282] S. Xu, Z. Ji, D. T. Pham, F. Yu, Bio-inspired binary bees algorithm for a two-level distribution optimisation problem, *Journal of Bionic Engineering* 7 (2010) 161–167.
- [283] S. K. Dhurandher, S. Misra, P. Pruthi, S. Singhal, S. Aggarwal, I. Woungang, Using bee algorithm for peer-to-peer file searching in mobile ad hoc networks, *Journal of Network and Computer Applications* 34 (2011) 1498–1508.
- [284] B. Segall, D. Arnold, Elvin has left the building: A publish/subscribe notification service with quenching, in: *Proceedings of AUUG97*, 1997, pp. 3–5.
- [285] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, T. D. Chandra, Matching events in a content-based subscription system, in: *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, PODC ’99*, ACM, New York, NY, USA, 1999, pp. 53–61.
- [286] A. Carzaniga, D. S. Rosenblum, A. L. Wolf, Design and evaluation of a wide-area event notification service, *ACM Trans. Comput. Syst.* 19 (2001) 332–383.
- [287] S. Sarat, V. Pappas, A. Terzis, On the use of anycast in dns, in: *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings. 15th International Conference on*, 2006, pp. 71–78.
- [288] T. Stevens, T. Wauters, C. Develder, F. De Turck, B. Dhoedt, P. Demeester, Analysis of an anycast based overlay system for scalable service discovery and execution, *Comput. Netw.* 54 (2010) 97–111.
- [289] J. Seo, K. Cho, W. Cho, G. Park, K. Han, A discovery scheme based on carrier sensing in self-organizing bluetooth low energy networks, *Journal of Network and Computer Applications* 65 (2016) 72 – 83.
- [290] R. Dissanayaka, S. K. Prasad, S. B. Navathe, V. Goyal, Bsi: Bloom filter-based semantic indexing for unstructured p2p networks, *International Journal of Peer to Peer Networks* 6 (2015) 11–30.
- [291] R. Lima, C. Baquero, H. Miranda, Adaptive broadcast cancellation query mechanism for unstructured networks, in: *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on*, 2015, pp. 176–181.
- [292] A. G. Medrano-Chávez, E. Pérez-Cortés, M. Lopez-Guerrero, A performance comparison of chord and kademia dhts in high churn scenarios, *Peer-to-Peer Networking and Applications* 8 (2015) 807–821.
- [293] M. Atif, M. Mousavi, Formal specification and analysis of accelerated heartbeat protocols, in: *Proceedings of the 2010 Summer Computer Simulation Conference, SCSC ’10*, Society for Computer Simulation International, San Diego, CA, USA, 2010, pp. 403–412.
- [294] E. Meshkova, J. Riihijarvi, P. Mahonen, Netp1-05: Evaluation of dynamic query abolishment methods in heterogeneous networks, in: *IEEE Globecom 2006*, 2006, pp. 1–6.
- [295] J. Zarrin, R. L. Aguiar, J. P. Barraca, Hard: Hybrid adaptive resource discovery for jungle computing, *Journal of Network and Computer Applications* 90 (2017) 42 – 73.
- [296] J. Zarrin, R. L. Aguiar, J. P. Barraca, Dynamic, scalable and flexible resource discovery for large-dimension many-core systems, *Future Generation Computer Systems* 53 (2015) 119 – 129.
- [297] J. Zarrin, R. L. Aguiar, J. P. Barraca, Elcore: Dynamic elastic resource management and discovery for future large-scale manycore enabled distributed systems, *Microprocessors and Microsystems* 46, Part B (2016) 221 – 239.
- [298] V. Nagarajan, M. Mohamed, A decentralized two phase resource discovery model for peer-to-peer grid environments, *Int J Adv Engg Tech/Vol. VII/Issue II/April-June VII* (2016) 1092–1095.
- [299] C. Pittaras, C. Papagianni, A. Leivadeas, P. Grosso, J. van der Ham, S. Papavassiliou, Resource discovery and allocation for federated virtualized infrastructures, *Future Generation Computer Systems* 42 (2015) 55 – 63.
- [300] Y. Zhang, Y. Jia, X. Huang, B. Zhou, J. Gu, A scalable method for efficient grid resource discovery, in: Y. Luo (Ed.), *Cooperative Design, Visualization, and Engineering*, volume 4674 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2007, pp. 97–103.
- [301] A. Iamnitchi, I. T. Foster, On fully decentralized resource discovery in grid environments, in: *Proceedings of the Second International Workshop on Grid Computing, GRID ’01*, Springer-Verlag, London, UK, UK, 2001, pp. 51–62.
- [302] C. N. Ververidis, G. C. Polyzos, Service discovery for mobile ad hoc networks: A survey of issues and techniques, *Commun. Surveys Tuts.* 10 (2008) 30–45.
- [303] M. H. Khoobkar, M. Mahdavi, Enabling efficient peer to peer resource discovery in dynamic grids using variable size routing indexes, in: *Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, ISPAN ’09*, IEEE Computer Society, Washington, DC, USA, 2009, pp. 691–695.
- [304] F. Butt, S. S. Bokhari, A. Abhari, A. Ferworn, Scalable grid resource discovery through distributed search, *arXiv*, arXiv:1110.1685 (2011).
- [305] A. Passarella, Review: A survey on content-centric technologies for the current internet: CDN and P2P solutions, *Comput. Commun.* 35 (2012) 1–32.
- [306] M. Klusch, Information agent technology for the internet: A survey, *Data Knowl. Eng.* 36 (2001) 337–372.
- [307] M. Hussin, N. A. W. A. Hamid, K. A. Kasmiran, Improving reliability in resource management through adaptive reinforcement learning for distributed systems, *Journal of Parallel and Distributed Computing* 75 (2015) 93 – 100.
- [308] A. Hameurlain, F. Morvan, Evolution of query optimization methods, in: *Transactions on Large-Scale Data- and Knowledge-Centered Systems I*, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 211–242.
- [309] S. Ghamri-Doudane, N. Agoulmine, Enhanced DHT-based P2P architecture for effective resource discovery and management, *J. Netw. Syst. Manage.* 15 (2007) 335–354.
- [310] D. Lazaro, J. Marques, X. Vilajosana, Flexible resource discovery for decentralized P2P and volunteer computing systems, in: *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, 2010 19th IEEE International Workshop on, 2010, pp. 235–240.
- [311] R. Baldoni, S. Bonomi, A. Cerocchi, L. Querzoni, Virtual tree: A robust architecture for interval valid queries in dynamic distributed systems, *J. Parallel Distrib. Comput.* 73 (2013) 1135–1145.
- [312] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. D. Kubiatowicz, Tapestry: A resilient global-scale overlay for service deployment, *Selected Areas in Communications, IEEE Journal on* 22 (2004) 41–53.
- [313] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, T. M. Gil, A performance vs. cost framework for evaluating DHT design tradeoffs under churn, in: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, 2005, pp. 225–236.
- [314] J. Risson, T. Moors, Survey of research towards robust peer-to-peer networks: Search methods, *Comput. Netw.* 50 (2006) 3485–3521.
- [315] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz, Handling churn in a DHT, in: *Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC ’04*, USENIX Association, Berkeley, CA, USA, 2004, pp. 10–10.

- [316] M. Castro, M. Costa, A. Rowstron, Performance and dependability of structured peer-to-peer overlays, in: Proceedings of the 2004 International Conference on Dependable Systems and Networks, DSN '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 9–18.
- [317] D. Liben-Nowell, H. Balakrishnan, D. Karger, Analysis of the evolution of peer-to-peer systems, in: Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing, PODC '02, ACM, New York, NY, USA, 2002, pp. 233–242.
- [318] S. Di, C. L. Wang, W. Zhang, L. Cheng, Probabilistic best-fit multi-dimensional range query in self-organizing cloud, in: 2011 International Conference on Parallel Processing, 2011, pp. 763–772.
- [319] J.-S. Kim, P. Keleher, M. Marsh, B. Bhattacharjee, A. Sussman, Using content-addressable networks for load balancing in desktop grids, in: Proceedings of the 16th International Symposium on High Performance Distributed Computing, HPDC '07, ACM, New York, NY, USA, 2007, pp. 189–198.
- [320] F. Falchi, C. Gennaro, P. Zezula, Nearest neighbor search in metric spaces through content-addressable networks, *Inf. Process. Manage.* 44 (2008) 411–429.
- [321] A. Datta, M. Hauswirth, R. John, R. Schmidt, K. Aberer, Range queries in trie-structured overlays, in: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, P2P '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 57–66.
- [322] S. Di, C.-L. Wang, W. Zhang, L. Cheng, Probabilistic best-fit multi-dimensional range query in self-organizing cloud, in: Proceedings of the 2011 International Conference on Parallel Processing, ICPP '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 763–772.
- [323] M. Cai, M. Frank, J. Chen, P. Szekely, Maan: A multi-attribute addressable network for grid information services, in: Proceedings of the 4th International Workshop on Grid Computing, GRID '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 184–191.
- [324] A. R. Bharambe, M. Agrawal, S. Seshan, Mercury: Supporting scalable multi-attribute range queries, *SIGCOMM Comput. Commun. Rev.* 34 (2004) 353–366.
- [325] D. Li, J. Cao, X. Lu, K. C. C. Chen, Efficient range query processing in peer-to-peer systems, *IEEE Transactions on Knowledge and Data Engineering* 21 (2009) 78–91.
- [326] P. Ganesan, B. Yang, H. Garcia-Molina, One torus to rule them all: Multi-dimensional queries in P2P systems, in: Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004, WebDB '04, ACM, New York, NY, USA, 2004, pp. 19–24.
- [327] A. González-Beltrán, P. Milligan, P. Sage, Range queries over skip tree graphs, *Comput. Commun.* 31 (2008) 358–374.
- [328] J. Albrecht, D. Oppenheimer, A. Vahdat, D. A. Patterson, Design and implementation trade-offs for wide-area resource discovery, *ACM Trans. Internet Technol.* 8 (2008) 18:1–18:44.
- [329] C. Zhang, W. Xiao, D. Tang, J. Tang, P2P-based multidimensional indexing methods: A survey, *J. Syst. Softw.* 84 (2011) 2348–2362.
- [330] F. Buccafurri, F. Furfaro, G. M. Mazzeo, D. Saccí, A quad-tree based multiresolution approach for two-dimensional summary data, *Inf. Syst.* 36 (2011) 1082–1103.
- [331] D. Chatziantoniou, A. Anagnostopoulos, Hierarchical stream aggregates: Querying nested stream sessions, in: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, SSDBM '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 439–.
- [332] D. Kostoulas, D. Psaltoulis, I. Gupta, K. P. Birman, A. J. Demers, Active and passive techniques for group size estimation in large-scale and dynamic distributed systems, *J. Syst. Softw.* 80 (2007) 1639–1658.
- [333] R. Van Renesse, K. P. Birman, W. Vogels, Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining, *ACM Trans. Comput. Syst.* 21 (2003) 164–206.
- [334] P. Cao, Z. Wang, Efficient top-k query calculation in distributed networks, in: Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing, PODC '04, ACM, New York, NY, USA, 2004, pp. 206–215.
- [335] P. Reynolds, A. Vahdat, Efficient peer-to-peer keyword searching, in: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Middleware '03, Springer-Verlag New York, Inc., New York, NY, USA, 2003, pp. 21–40.
- [336] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker, Making gnutella-like P2P systems scalable, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, ACM, New York, NY, USA, 2003, pp. 407–418.
- [337] A. S. Tigelaar, D. Hiemstra, D. Trieschnigg, Peer-to-peer information retrieval: An overview, *ACM Trans. Inf. Syst.* 30 (2012) 9:1–9:34.
- [338] S. Ghamri-Doudane, N. Agoulmine, Enhancing the P2P protocols to support advanced multi-keyword queries, in: Proceedings of the 5th International IFIP-TC6 Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, NETWORKING'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 630–641.
- [339] M. Steiner, E. W. Biersack, T. En Najjary, Actively monitoring peers in KAD, in: IPTPS 2007, 6th International Workshop on Peer-to-Peer Systems, February, 2007, Bellevue, USA, Bellevue, UNITED STATES, 2007, pp. 26–27.
- [340] D. Carra, E. W. Biersack, Building a reliable P2P system out of unreliable P2P clients: The case of kad, in: Proceedings of the 2007 ACM CoNEXT Conference, CoNEXT '07, ACM, New York, NY, USA, 2007, pp. 28:1–28:12.
- [341] M. Steiner, T. En-Najjary, E. W. Biersack, Long term study of peer behavior in the kad DHT, *IEEE/ACM Trans. Netw.* 17 (2009) 1371–1384.
- [342] J. Li, B. Loo, J. Hellerstein, M. Kaashoek, D. Karger, R. Morris, On the feasibility of peer-to-peer web indexing and search, in: Peer-to-Peer Systems II, volume 2735 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2003, pp. 207–215.
- [343] Z. Gao, X. Li, L. Wang, J. Zhao, Y. Zhao, H. Shi, BfgsdP: Bloom filter guided service discovery protocol for MANETs, in: Proceedings of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks, ITC20'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 446–457.
- [344] S. Cheng, C. K. Chang, L.-J. Zhang, An efficient service discovery algorithm for counting bloom filter-based service registry, in: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 157–164.
- [345] D. Oppenheimer, J. Albrecht, D. Patterson, A. Vahdat, Distributed resource discovery on planetlab with SWORD, in: Proceedings of the ACM/USENIX Workshop on Real, Large Distributed Systems (WORLDS), 2004.
- [346] D. Oppenheimer, J. Albrecht, D. Patterson, A. Vahdat, Design and implementation tradeoffs for wide-area resource discovery, in: High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium on, 2005, pp. 113–124.
- [347] H. M. N. D. Bandara, A. P. Jayasumana, Characteristics of multi-attribute resources/queries and implications on P2P resource discovery, in: Proceedings of the 2011 9th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 173–180.
- [348] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: 2008 Grid Computing Environments Workshop, 2008, pp. 1–10.
- [349] K. Sathish, A. RamaMohan Reddy, Workflow scheduling in grid computing environment using a hybrid gaaco approach, *Journal of The Institution of Engineers (India): Series B* (2016) 1–8.
- [350] H. B. Prajapati, V. A. Shah, Scheduling in grid computing environment, in: 2014 Fourth International Conference on Advanced Computing & Communication Technologies, 2014, pp. 315–324.
- [351] L. Schubert, A. Kipp, Principles of service oriented operating systems, in: P. Vicat-Blanc Primet, T. Kudoh, J. Mambretti (Eds.), *Networks for Grid Applications*, volume 2 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, 2009, pp. 56–69.
- [352] The S[o]OS Consortium , S(o)OS (Service-oriented Operating System): Resource-independent execution support on exa-scale systems, <http://www.soos-project.eu/>, 2010-2013. [Online: accessed 9-January-2017].
- [353] J. Zarrin, R. L. Aguiar, J. P. Barraca, Manycore simulation for peta-scale system design: Motivation, tools, challenges and prospects, *Simulation Modelling Practice and Theory* 72 (2017) 168 – 201.
- [354] J. Zarrin, R. L. Aguiar, J. P. Barraca, A specification-based anycast

- scheme for scalable resource discovery in distributed systems, in: 10th ConfTele 2015 - Conference on Telecommunications, Sep 2015, pp. 13–17.
- [355] J. Zarrin, R. L. Aguiar, J. P. Barraca, A self-organizing and self-configuration algorithm for resource management in service-oriented systems, in: 19th IEEE Symposium on Computers and Communications (IEEE ISCC 2014), Madeira, Portugal, 2014, pp. 1–7.
- [356] C. P. R. Baaij, J. Kuper, L. Schubert, SoOSiM: Operating System and Programming Language Exploration, in: G. Lipari, T. Cucinotta (Eds.), Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time System (WATERS), 2012, pp. 63–68.
- [357] G. Lipari, E. Bini, A framework for hierarchical scheduling on multi-processors: From application requirements to run-time allocation, in: Real-Time Systems Symposium (RTSS), 2010 IEEE 31st, 2010, pp. 249–258.
- [358] T. Cucinotta, Challenges in operating system design for future many-core systems, All Hands Meeting (AHM) 2010, Cardiff, UK, Available at <http://retis.sssup.it/~tommaso/presentations/AHM-2010.pdf>, 2010. [Online: accessed 15-April-2016].
- [359] L. Schubert, Dynamicity requirements in future cloud-like infrastructures, Invited Speaker, EuroCloud CLASS Conference, Available at http://videlectures.net/classconference2012_schubert_infrastructures/, 2012. [Online: accessed 9-January-2017].
- [360] L. Schubert, A. Kipp, S. Wesner, Above the clouds: From grids to service-oriented operating systems., in: Future Internet Assembly, 2009, pp. 238–249.
- [361] The Barrelfish OS c/o ETH Zurich, Microsoft Research in Cambridge, The barrelfish operating system, Available at <http://www.barrelfish.org/>, <http://www.barrelfish.org/documentation.html>, 2009-2016. [Online: accessed 14-February-2017].
- [362] Institute of Information Resource Management, Ulm University, Germany, Mythos: Many threads operating system, Available at <https://www.uni-ulm.de/en/in/omi/research/research-projects/mythos/>, 2013-2017. [Online: accessed 14-February-2017].
- [363] R. Aguiar, D. Gomes, J. Barraca, N. Lau, Cloudthinking as an intelligent infrastructure for mobile robotics, Wireless Personal Communications 76 (2014) 231–244.