# P2P4GS: A Specification for Services Management in Peer-to-Peer Grids

Bassirou Gueye and Ibrahima Niang
*Département de Mathématiques et d'Informatique*
*Université Cheikh Anta Diop*
*Dakar, Senegal*
*Email: {bassirou.gueye, ibrahima1.niang}@ucad.edu.sn*

Olivier Flauzac and Cyril Rabat
*CReSTIC, UFR Sciences Exactes et Naturelles*
*Université de Reims Champagne Ardenne*
*Reims, France*
*Email: {olivier.flauzac, cyril.rabat}@univ-reims.fr*

***Abstract*—The grid-based peer-to-peer architectures were used either for storage, data sharing and computing. So far, the proposed solutions of grid services are generally based on hierarchical topologies, which present a high degree of centralization. The main issue of this centralization is the unified management of resources. Therefore, it is difficult to react rapidly against failures that can affect end-users. In this paper, we propose an original specification, called P2P4GS, that enables self-managed services in peer-to-peer grids. The objective is to design a self-adaptive solution allowing services deployment and invocation based on the paradigm of peer-to-peer services. These tasks are completely delegated to the platform and are achieved through a transparent manner to the end-user. The proposed specification is not linked to a fixed peer-to-peer architecture or to a services management protocol. Furthermore, we propose a detailed illustration of our P2P4GS specification.**

***Keywords-Peer-to-peer network; Grid computing; Web Services; Information Systems.***

## I. INTRODUCTION

Grid Computing is a technology which aims to offer to virtual organizations and scientific community virtually unlimited computing resources [1][2]. The goal of such a paradigm is to enable to virtual organizations (VO) the federated resource sharing in dynamic and distributed environments. Indeed, end-users can access large computing and storage resources that they could not operate otherwise.

The emergence of Web Services [3] provided a framework that initiated its alignment with the grid computing technologies as well as cloud computing [4]. These convergences allowed the appearance, on one hand, of grid services [5][6] and on the other hand, of cloud services [7][8].

Indeed, grid services are the result of research established by the OGF (*Open Grid Forum*) and leading to the OGSA (*Open Grid Service Architecture*) [5] and the OGSI (*Open Grid Service Infrastructure*) [6]. These grid services enable to use resources more rationally. This is due to leveraging load-balancing mechanism between nodes. A particular effort was made in order to normalise the grids services. Like Web Services, the goal is to gather resources in order to promote the development of applications that use grid based services.

Notwithstanding, grids that use the concept of services are generally based on highly centralized hierarchical architectures [2][9]. This centralization involves an unified management of resources as well as a difficulty to react promptly against failures and faults that affect the community (the community designates the set of nodes of the grid).

To overcome these drawbacks, convergence solutions of grid computing and peer-to-peer systems have been proposed [10][11][12][13]. However, a semantics of the services execution chain does not exist in the literature. Most of proposed solutions around peer-to-peer grids are limited on resource discovery and are generally distinguished by the type of search algorithm used.

The main goal of this paper is to present a new specification of services management in a grid computing based on peer-to-peer architecture. This specification, called P2P4GS (*Peer-To-Peer For Grid Services*), presents the originality to not dissociate the peer-to-peer infrastructure to the service management platform. Indeed, we propose to separate the peer-to-peer grid management layer to the location and runtime services layer. Since the proposed specification is generic, any combination of peer-to-peer protocol in accordance with the constraints of our model could be composed with any services execution platform. Deployment as well as invocation is absolutely delegated to the platform. Consequently, both tasks become transparent to the end-user. In fact, the end-user can be a simple user who has no knowledge of the system (a student or a researcher for instance) or an administrator of grid services.

The remainder of the paper is organized as follows. Firstly, Section II is devoted to present an overview on the related work. Next, in Section III, we describe the model that we use, as well as different aspects of the P2P4GS specification. Subsequently, we propose a detailed illustration of our specification in Section IV. Finally, Section V concludes the paper and outlines our future works.

## II. RELATED WORK

The remote code execution has been defined in various manners. The possibility to remotely call a procedure led to the concept of RPC (*Remote Procedure Call*). RPC specification specifies how information is exchanged between a

 ISBN: 978-1-61208-365-0

consumer (client) and a resource (server). Different implementations that consider different languages, protocols and systems have been proposed. For instance, we can cite the following implementations based on Web Services: XML-RPC, SOAP [3]. RPC concept has been transferred in the grid either from libraries that enable technical solutions, like Globus [14], or directly from the solutions that are designed to ensure grid RPC. However, the proposed solutions are hierarchical and need centralization points that gather the related information to a given service. For example, tools such as DIET [9] should register their requirements (e.g., computational or memory resource, data resource, applications, etc.) on a fixed centralization point.

In addition, in the case of Web Service execution on the Internet, solutions of deployment [6] and resource discovery [15][16] in such environments have been proposed. In fact, the authors of [15] proposed a hierarchical topology of web services for resource discovery in Grids based on Globus Toolkit. In this proposed solution, the Grid system is divided into virtual organizations (VO). On the other hand, Fouad et al. [16] presented scalable Grid resource discovery through a distributed search. The proposed model includes the application layer which provides a web interface for the user and the collective layer which is a web service to discover resources. The resource discovery model contains the metadata and resource finder web services to provide a scalable solution for information administrative requirements when the grid system expands over the Internet.

Nevertheless, these works are based on solutions with hierarchical architectures that have a very low degree of dynamicity and whose infrastructural services are themselves centralized. Indeed, these centralized or hierarchical approaches suffer from a single point of failure, of bottlenecks in highly dynamic systems, and lack of scalability in highly distributed environments [17].

Fortunately, grid and P2P systems share several features and can profitably be integrated. Convergence of the grid system with the philosophy and techniques of the P2P architecture is a promising approach to alleviate the disadvantages of traditional grid systems. Torkestani [18] asserted that P2P Grids exploit the synergy between the Grid system and peer-to-peer network to efficiently manage the Grid resources and services in large-scale distributed environments. The decentralized approach can enhance scalability and fault-tolerance but it induces a very large network traffic and may limit search effectiveness. In addition, Marin Perez et al. [19] argue that the ultimate goal of building P2P Grid is to integrate the P2P, Grid, and Web Services.

However, most of proposed solutions with respect to peer-to-peer grids are limited to the resource discovery [10][11][12][13][18] and are generally distinguished by the type of search algorithm used. The authors of the survey [20] argue that the resource discovery (which aims to discover appropriate resources based on a requested task) is one of the essential challenges in Grid. There are certain factors that make the resource discovery problem difficult to solve. For instance, Torkestani [13] proposed a multi-attribute distributed learning automata-based resource discovery algorithm for large-scale peer-to-peer grids. Based on the learning automata theory, the author proposed a method that enables to route the resource query through the path having the minimum expected hop count toward the grid peers including the requested resources.

## III. Overview of the P2P4GS specification

Initially, we defined the concepts related to the various components of our systems. Then, in order to present our specification, we describe the application model.

### A. Preliminaries

*1) Node concept:* a node is a machine (with acceptable computational power) or a site (cluster). We assume that each node has a unique identifier in the communication network. Nodes are in charge of the local management of the network. Therefore, they collectively provide the tasks described in Section III-B. Moreover, they manage the execution platform, which ensures the following tasks: the deployment, the life cycle of services, the management of requests and their executions. In addition, each node maintains a table called "*Service Registry*", which lists the services owned by this node, as well as the other services located inside the grid and learn during a discovery process.

*2) Service concept:* the services can be viewed as different objects that are integrated to the runtime platforms located in each node. A service is characterized by:

- the implementation platform;
- the required resources for its execution (computational power, data resource and connection resource, etc.);
- the format and constraints according to the requested invocation and obtained results.

### B. Modelization

We define an architecture model composed of four layers. This enables us to define each layer independently. Indeed, with this model, each layer solves a number of problems (handles a number of tasks required by the overall system), in order to provide well-defined services to the higher layers. Figure 1 illustrates the architecture of our specification. In the following, we describe the different layers of the architecture.

*1) Physical layer:* this layer provides addressing, routing and communication functions. We are modeling this layer as a directed and connected graph $G_1 = (V, E_1)$ where $V$ is the set of nodes that host the services and $E_1$ the set of directed communications links between the nodes.

These definitions allow taking into account the different characteristics of the network:

 ISBN: 978-1-61208-365-0 

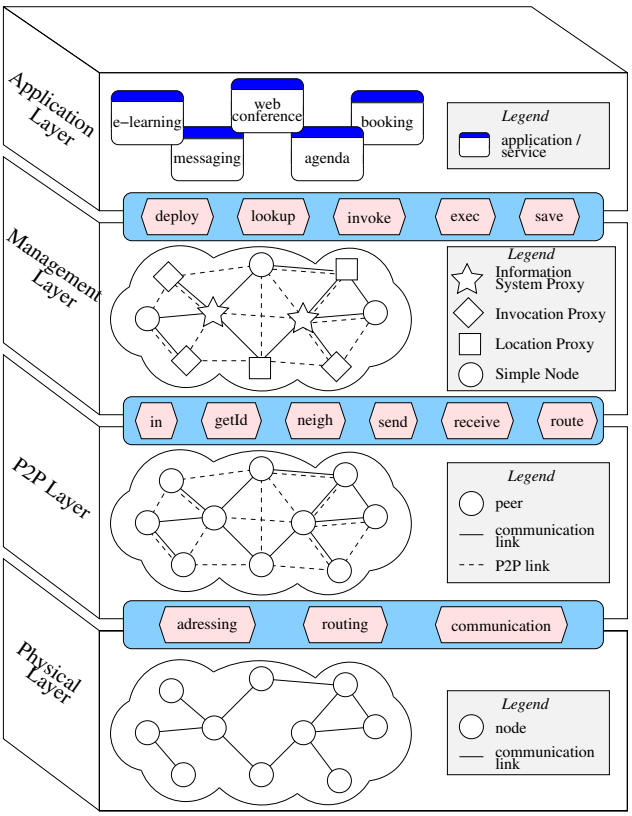


Figure 1: Architecture of P2P4GS specification

- security elements such as firewalls that restrict the possibility of communication of the nodes;
- configuration and implementation elements of the network such as the NAT.

*2) P2P layer:* it corresponds to the peer-to-peer middleware and provides communication functionalities and maintenance of the topology. P2P layer is modeled as an undirected and connected graph $G_2 = (V, E_2)$ where $V$ corresponds to the set of nodes that hosting the services and $E_1$ the set of undirected communications links between the nodes established by the P2P protocol.

We assume that this layer offers at the minimum the following primitives :

- **in()** : this primitive is executed by a new node that is connected to the P2P community;
- **getId()** : returns the P2P node's identifier;
- **neigh()** : retrieves the list of all its P2P neighbors;
- **send(id, message)** : sends a message to a node identified by "id" in the system;
- **receive()** : receives a message from a node of the system;
- **route(id, message)** : routes a message to a destination.

Therefore, any peer-to-peer system presenting these basic features can be exploited by our specification.

*3) Management layer:* this layer constitutes the core of our specification. All tasks of management, administration and maintenance of a service are defined in this layer. These tasks ensure the management of life cycle of a given service. Indeed, P2P4GS specification provides a set of primitives that are "*deploy*", "*lookup*", "*invoke*", "*exec*" and "*save*" for respectively deployment, location, invocation, execution and recording of a service. These primitives are described in Section III-C.

Given that the size of distributed systems grows in terms of number of nodes, services and users, in order to ensure scalability, we propose to limit the knowledge on some nodes that we call *proxies*. In fact, we define three types of proxies: the ISP *(Information System Proxy)*, the IP *(Invocation Proxy)* and the LP *(Location Proxy)* for effectively ensure function of the primitives above mentioned. According to the P2P paradigm, the status of the nodes is not fixed and can evolve during execution. Note that, the default node status is *Simple Node*. The different proxy nodes are described in Section III-D.

*4) Application layer:* this layer is the upper layer that interfaces with the end-users. Primitives of the underlying layer are exploited by different platforms with whom they interact in order to provide services to Application layer. Note that, the access to P2P grid resources will be done in a transparent manner with respect to the end-user.

### C. Services specification in P2P environment

In this section, we define the set of primitives and specific operations to the services runtime on our P2P grid system.

*1) Service deployment ("deploy"):* the service deployment requires the detection of nodes that are able to host and execute a given service. A subset of nodes of the community will be candidates for hosting the service. The following strategies can be used in order to select the node that hosts the service among the candidate nodes:

1) a *random strategy* based on a random selection of a node where the service will be deployed on;
2) a *balanced strategy* based on statistic data gathering about nodes, and selection of the less loaded nodes where the service can be deployed on;
3) a *first node strategy* based on exploration: the first node able to support the service is selected.

**Remark 1.** *The first strategy needs to know all candidate nodes for deployment; the second one to know all candidate nodes and information on each; the third limits the required knowledge and also reduces the communications costs but can cause problems of load-balancing.*

*2) Service location ("lookup"):* the service location is the first step of a service execution process. Each node of the peer-to-peer grid has a table called *Service Registry* that stores node's identifier of the grid according to their signatures. "*lookup*" can be considered as a local service.

 ISBN: 978-1-61208-365-0

Indeed, it is available on each node of the community, but cannot be invoked from a remote machine.

Any node of the P2P4GS community that receives a service location request, invokes the "*lookup*" function which performs the following operations:

- if the service is located on the node, this latter returns its identifier and information about the service (e.g., binding, invocation constraints, ...);
- if the service is not located on the node, then :
  - if the node knows the location of service, it routes the request to the node that hosts it;
  - if the node does not know the location of service, it forwards the request to its neighbors.

**Remark 2.** *The required service may be missing within the grid, whether due to a failure, otherwise because it has not been yet deployed. In this case, we can use an algorithm like PIF (Propagation of Information and Feedback), which returns to the entry point (III-C3) an error code that will be sent to the sender of the request.*

*3) Service invocation ("invoke"):* the invocation of a given service can be achieved from any node of the community. In the case where an outside node of community wishes to invoke a given service, it passes through a node of the community which becomes "entry point" and thus its "*invocation point*". This entry point is in charge of making the invocation of the service, retrieves the result and returns it to the request source.

**Remark 3.** *A node "entry point" is a node from which an end-user logs in to access the community. In fact, the system provides to the end-user a kind of black box in which all tasks will be done in a transparent manner. Note that any node of the community can serve as an "entry point".*

*4) Service execution ("exec"):* once the request has reached the node that keeps the service, it is executed. The first step is to identify the required invocation parameters for the execution. The execution can either produce the result, or trigger an error. Once the result has been produced, it is routed to the "invocation point".

*5) Service recording ("save"):* it is possible to exploit the platform through research without memory of services on each invocation. As we stated previously, we can increase the performance of our platform with distributed service registries. We plan the construction of these registries according to two main strategies:

- *Flooding deployment*: in this first strategy, the diffusion of the information on the service location will be achieved according to a flooding approach.
- *Broadcast-based on response invocation*: in this second strategy, the location information will be broadcast after the response of the invocation. Each node that relays the response towards the invocation point will store the information on the service location.

**Remark 4.** *In order to avoid a node overloading-memory by the service register, it is possible to aggregate information on several consecutive nodes.*

### D. Specification of the node proxy concept

As our specification is not related to a particular peer-to-peer architecture, we propose an organizational model of nodes in the system which tends to limit the knowledge on some nodes called proxies in order to avoid an overloading-memory of the community nodes. In fact, we propose to limit knowledge about the location on some nodes in the system (as well as their list of services) on nodes we call ISP *(Information System Proxy)*.

An ISP acts as an information system for a set of nodes. Consequently, it knows the location of a certain number of services in the community.

In order to limit the ISP overload, we delegate invocation and execution services tasks for nodes called IP *(Invocation Proxy)*. Moreover, services have not the same execution constraints in terms of CPU, RAM, execution platform, etc., Thus, a node is defined as IP for a given service, if it knows its location and respects its implementation constraints. Therefore, IP node owns the client part of this service (stub).

A node having only a knowledge about the location of a given service but not having its stub will be called LP *(Location Proxy)* for this service.

It should be noted that a node is both invocation proxy and location proxy for its own services.

**Remark 5.** *In order to avoid the overload of the memory of the ISP, IP and LP nodes, we propose to remove the knowledge about a given service at the end of a TTL (Time-To-Live). Accordingly, a service that is rarely requested will not be stored indefinitely. In contrast, a node will be always an invocation proxy for a service frequently requested of the fact that its TTL will be reset after each new invocation. In the case of the ISP nodes, this TTL is only applicable on services known through other ISP nodes.*

## IV. Management of the P2P4GS community nodes

In this section, we describe a detailed illustration of our specification. We firstly present the creation process of the P2P4GS community. Afterwards, we describe the location and deployment processes of a given service in this community.

### A. Organization of the P2P4GS community

Given that nodes of a grid present minimum capacities in terms of CPU, memory and bandwidth, any node can potentially become ISP. Thus, no election of ISP node is necessary. Moreover, in a large-scale environment where nodes are geographically dispersed, the network does not formed spontaneously. Therefore, using the concept of direct neighborhood, we propose a progressive ISP election

 ISBN: 978-1-61208-365-0 

according to node's connection. We assume that direct neighborhood of a node is established upon first connection and can evolve during execution.

We define the concept of direct neighborhood as follows: for any pair of nodes ($u$, $v$) of the P2P4SG community, $u$ and $v$ are direct neighbors if and only if:

- $u$ can communicate with $v$ and *vice-versa*. This means that there exists a bidirectional communication link between $u$ and $v$;
- $u$ knows $v$ in its list of neighbors and vice-versa.

The following steps describe the process creation of P2P4GS community:

- The first node connected to the system (then it does not have any neighbors), elects itself as ISP ;
- Any new node that connects to the system checks its neighborhood table:
  - If it has at least one ISP as neighbor then it keeps its status (simple node). Moreover, if it has hosted services, it sends information on services to its ISP that will update their *Service Registry* ;
  - Otherwise, it elects itself as ISP and informs its neighbors.

### *B. Elasticity management in the P2P4GS community*

As large scale systems can be highly dynamic, disconnection of a node can occur at any time. In order to ensure scalability, it is necessary to take into account the disconnection of the nodes. Thus, according to the disconnected node's status, we propose these following approaches.

- Disconnection of a simple node: in this case, its services become unavailable. As noted in Remark 5, if a service remains unavailable until end of the TTL, its knowledge will be deleted at the different service registries having recorded it.
- Disconnection of a LP node: in this case, source node will start a new process of discovery if it does not know another LP for desired service.
- Disconnection of an IP node: same treatment as the case of a disconnected LP.
- Disconnection of an ISP node: in this case, one of the following events occurs:
  - In the case of the service location, processing remains the same as those IP or LP node;
  - In the case of the service deployment, if candidate node has other ISP neighbors, it sends services information to its ISP neighbors that will update their *Service Registry*. Otherwise, it elects itself as ISP and informs its neighbors.

### *C. Location and deployment process in the P2P4GS community*

In this section, we first describe the service location process by using the example shown in Figure 2. Subsequently, we explain the deployment process.

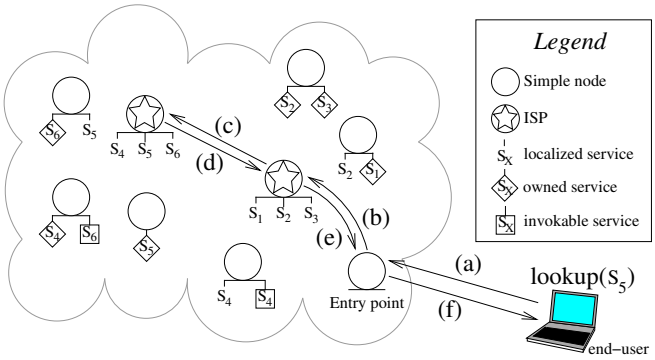


Figure 2: Example of location process in P2P4GS community

In this example, the *end-user* is connected to its *entry point* and the requested service is named $S_5$ ($a$). If the *entry point* knows the location of the requested service, it directly responds to the *end-user* by sending the corresponding service address. Else, if the node does not know the location of the service, it forwards the request to its ISP neighbors ($b$).

Any ISP that receives the location request consults its registry and performs a following tasks:

- if the service is registered, it responds to the source node by sending the corresponding service address;
- otherwise, we propose two strategies in order to explore others node's community:
  - construction of a structured overlay formed by ISP;
  - utilization of a wave algorithm such as the PIF for example.

The first strategy allows ISP nodes to communicate to each other in order to facilitate the processing of location request. However, this method induces costs for maintaining the overlay. The second limits the necessary knowledge therefore it is simpler to implement. However, it presents a non-negligible communication cost.

In this example, we have used the structured overlay formed by ISP nodes ($c$). According to the used approach, if the service is found, the response takes the opposite path of the location request. By using the "*save*" primitive, all intermediate ISP and the *entry point* node will update their service registry ($d$, $e$, $f$). Note that, as we have showed in Remark 5, a TTL is applied on this knowledge.

In the case of the deployment of a new service within the community, one of the strategies (random, balanced or first node) proposed in Section IV.C.1 can be applied. Note that for these strategies, the election process of node candidate keeps typically the same approach as for the location of service. The request will be based on the execution constraints (CPU, memory resource, execution platform, etc.) of the service to be deployed. Once the deployment is successfully completed, the candidate node sends to its ISP neighbors the information on the new hosted service.

 ISBN: 978-1-61208-365-0 

## V. Conclusion and future work

In this paper, we proposed an original specification of a "self-managed service in peer-to-peer grid" named P2P4GS. This specification is generic, not linked to a fixed peer-to-peer architecture or a given services management protocol. Given that the size of distributed systems grows in terms of number of nodes, services and users, we have proposed to limit the knowledge on some nodes that we have called *proxies*, in order to ensure scalability. After having described our model, as well as different aspects of the P2P4GS specification, we have presented a detailed illustration of this specification.

Currently, we are working on the validation of our specification under *Oversim framework* based on *OMNeT++* simulator in order to evaluate: service's deployment and location in each strategy, communication cost in terms of messages and fault tolerance mechanisms.

As future work, we plan to implement and compare our specification with other existing solutions.

## References


[1] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," in *Peer-to-Peer Systems II*. Springer, 2003, pp. 118–128.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web services - Concepts, Architectures and Applications*. Springer Verlag, ISBN 3-540-44008-9, chapter 5, 2004.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam *et al.*, "The open grid services architecture (ogsa), version 1.5." OGF Specification GFD-I. 080, July 2006.

[6] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, "Open grid services infrastructure (ogsi), version 1.0," June 2003.

[7] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (iaas)," *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60–63, 2010.

[8] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 159–168.

[9] E. Caron and F. Desprez, "Diet: A scalable toolbox to build network enabled servers on the grid," *International Journal of High Performance Computing Applications*, vol. 20, no. 3, pp. 335–352, 2006.

[10] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-peer resource discovery in grids: Models and systems," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 864–878, 2007.

[11] T. Kocak and D. Lacks, "Design and analysis of a distributed grid resource discovery protocol," *Cluster Computing*, vol. 15, no. 1, pp. 37–52, 2012.

[12] D. Chen, G. Chang, X. Zheng, D. Sun, J. Li, and X. Wang, "A novel p2p based grid resource discovery model," *Journal of Networks*, vol. 6, no. 10, pp. 1390–1397, 2011.

[13] J. A. Torkestani, "A multi-attribute resource discovery algorithm for peer-to-peer grids," *Applied Artificial Intelligence*, vol. 27, no. 7, pp. 575–598, 2013.

[14] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Journal of computer science and technology*, vol. 21, no. 4, pp. 513–520, 2006.

[15] T. Gomes Ramos and A. C. M. A. de Melo, "An extensible resource discovery mechanism for grid computing environments," in *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, vol. 1. IEEE, 2006, pp. 115–122.

[16] F. Butt, S. S. Bokhari, A. Abhari, and A. Ferworn, "Scalable grid resource discovery through distributed search," *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 2, no. 5, pp. 1–19, 2011.

[17] Y. Deng, F. Wang, and A. Ciura, "Ant colony optimization inspired resource discovery in p2p grid systems," *The Journal of Supercomputing*, vol. 49, no. 1, pp. 4–21, 2009.

[18] J. A. Torkestani, "A distributed resource discovery algorithm for p2p grids," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 2028–2036, 2012.

[19] J. M. Marin Perez, J. B. Bernabe, J. M. Alcaraz Calero, F. J. Garcia Clemente, G. M. Perez, and A. F. Gomez Skarmeta, "Semantic-based authorization architecture for grid," *Future Generation Computer Systems*, vol. 27, pp. 40–55, 2011.

[20] N. Jafari Navimipour, A. Masoud Rahmani, A. Habibizad Navin, and M. Hosseinzadeh, "Resource discovery mechanisms in grid systems: A survey," *Journal of Network and Computer Applications*, pp. 389–410, 2013.


 ISBN: 978-1-61208-365-0 