A Scalable Hybrid Architecture based SKOS Ontology for Resource Discovery in Grid

Nabila Chergui
MISC laboratory
Mentouri University
Constantine, Algeria
chergui.nabila@gmail.com

Salim Chikhi
MISC laboratory
Mentouri University
Constantine, Algeria
slchikhi@yahoo.com

***Abstract*—Resource discovery is an important service in a grid, since a grid enables the sharing and collaborating of a wide variety of resources which should be fully exploited. Based on Semantic Web technologies and the idea of resources clustering, this paper presents a comprehensive hybrid architecture based on SKOS lightweight ontology to organize and discover resources in an efficient and scalable manner. Nodes with the same domain of interest get together into collections called federations in a hybrid grid, in which intra-federation adopts centralized management and inter-federation form a distributed one. The architecture strikes a balance between high efficiency of total centralized management and good scalability of distributed disposal. We propose an efficient process of query routing between semantically related federations, which decreases the cost of resource discovery, and an efficient and fault tolerant mechanism to leader's federation election that minimizes the overhead between leaders during the process of updating information about the other leaders' federations.**

Keywords-component: Resource discovery, SKOS, Hybrid architecture, Semantic query processing.

I. INTRODUCTION

Grid Computing is a virtualized distributed environment, aimed at enabling the sharing of geographically spread resources. Resource discovery mechanism is one of the fundamental requirements for grid computing systems, as it aids in resource management and applications scheduling. Resource discovery involves searching for resources that match the user's applications' requirement. An efficient resource discovery mechanism depends on two factors: first, the architecture of the system, hence the mechanism used to query processing across this architecture; second, the structure used to represent resources and consequently, the structure of queries.

In this work, we focus on the first factor of resource discovery. In this issue, various kinds of solutions to grid resource discovery have been proposed, the centralized one is efficient, although has some shortages as the central server in centralized architecture has a single point of failure, and it can create a bottleneck due to the sent messages to a single point, which may render such a system poor scalability.

The introduction of P2P and DHT techniques into grids brings some benefits like adaptation, self organization, fault tolerance; however, they are less efficient and have several limitations such as a risk of network congestion and overhead due to the sent messages for updating dynamic data on resources, and while searching for a resource, and the risk of churn effect if a large number of nodes want to update their data at the same time.

To address these problems, we propose a mechanism based on semantic nodes clustering into federations using a **SKOS***(Simple Knowledge Organization Scheme)* [1] lightweight ontology to regroup nodes having the same domain of interest and to process queries between them, which performs an effective searching according to the semantic distribution of nodes into federations and thus their resources. SKOS compared to OWL [2], has the ability to describe synonyms and associative relationships, to add information to concepts which can be easily used for defining ontological terminologies and enriched them by supplementary information about context. We will use a hybrid architecture composed of three layers, which combines the advantages offered by the above types of architectures. Clustering, is the most popular technique for creation of hybrid overlay networks, the main aim of clustering is to keep such desirable properties of distributed and centralized architectures [3]. Our approach supports a semantic organization of nodes in a hybrid way, and used semantic to handle queries between federations. By the integration of Semantic Web techniques and hybrid architecture, this approach speeds up the information query, and it guaranties, the scalability and the flexibility of the system.

By analogy to the real-world, countries are gathered in federations according to their interests. Under one federation, these last works on collaboration, share their resources to achieve their objectives. We draw inspiration from this to organize nodes. Nodes in the grid environment correspond to countries on the real-world, each node has a domain of interest such as mathematics, biology; to construct our federations we need to extract the domain of interest from node, then a measure of semantic similarity is applied between node and concepts in a SKOS lightweight ontology, to affect the node to its adequate federation.

The remainder of this paper is organized as follows:

Section II presents related works in this topic. Section III provides in a clear manner an explanation of the system, the construction of semantic federations and the three layered architecture based on SKOS. Section IV proposes algorithms

to explain how query processing is semantically performed, the leader is elected and federations are maintained. Theoretical performance evaluation is given in Section V. Conclusion and future works are provided in Section VI.

II. RELATED WORKS

In the literature, it exists many different approaches addressing the problem of resource discovery on grid environment. We can classify them on non semantic approaches, and semantic ones.

In non semantic approaches we find, centralized techniques like Condor [4], which uses a matchmaker with a centre server to process queries; it has a single point of failure and scale poorly. P2P techniques like [5] organize information nodes into a flat unstructured P2P network and random-walk based methods are used for query forwarding. Random-walks are not efficient in response time for a very large system. [6] Proposes a hierarchical structure to organize information nodes to reduce redundant messages. However, a well-defined hierarchy does not always exist, and the global hierarchy is hard to maintain in a dynamic environment.

Semantic techniques are those that use Semantic Web technologies. Semantic Web [7] attempts to define the metadata information model for the World Wide Web to aid in information retrieval and aggregation. Currently, many P2P applications have leveraged Semantic Web technologies to add semantics to P2P systems. Edutella [8] is a P2P network for searching Semantic Web metadata. Each Edutella peer can make its metadata information available as a set of RDF statements. The distributed individual RDF peers register the queries they may be asked through the query service, and queries are sent through the Edutella network to the subset of peers who have registered with the service to be interested in this kind of query. To forward queries between nodes, Edutella uses JXTA to broadcast queries to a HyperCup topology. Similarly, [9, 10] use broadcast/flooding to search semantic metadata. The simple P2P broadcast structure used by these systems makes them very difficult to scale to large-scale networks. Our system solves this problem by topology adaptation and semantics-based routing.

Semantic clustering or semantic hybrid approaches have appeared with the idea of grouping nodes with similar contents together to facilitate searching. [11, 12] Use a centralized server or super-peers to cluster nodes. However, the efficient communication mechanism between super-peers is absent in these systems. [13] Proposes to cluster nodes with similar interest together into communities, without discussing how to define the interest similarity among peers and how to form clusters. [14, 15] Add semantic short-cuts to group nodes. The short-cut approach relies on the presence of interest-based locality. Each peer builds a shortcut list of nodes that answered previous queries. To find content, a peer first queries the nodes on its shortcut list and only if unsuccessful, floods the query. [16] Uses semantic clustering to organize the network topology and reduce search space to semantically related clusters; instead it uses a complex and costly mechanism to construct clusters each time it needs to add new node to the system.

III. OVERVIEW OF THE SYSTEM

This section illustrates how to provide efficient construction of federations, and gives a detailed explanation of the system architecture.

A. Semantic Federations Construction Based on SKOS Ontology Domain Description

As we have mentioned above, the computing grids can create federations in scientific domains, such as physics, earth science and so on; each federation is formed by a collection of nodes with the same domain of interest because we believe that more nodes shares the same interest, more their resources tend to be similar. A federation is managed by a leader and consists of members that serve as workers. Communication and collaboration can operate on top of the federations. With federations, grid users can easily share resources and knowledge within the federation.

To create grid federations, we need a classification technique to classify nodes. Since each node has a specific domain of interest, **Ont**ology of **D**omains **D**escription **OntDD** is used to classify grid domain applications in general. This ontology is a lightweight ontology; we used SKOS [1] vocabulary, SkosEd editor [17], Skos API [18] and Protégé 4 [19] to formalize it.

SKOS Used to represent term lists and controlled vocabularies, to provide a simple machine-understandable. Technologies such as RDF and OWL are seen as key elements for building a Semantic Web. The SKOS model is built in accordance with these technologies and has a serialization to the Resource Description Framework (RDF). (See Table I for an example). In general, KOS differs significantly from formal ontologies represented using OWL, as they do not contain detailed intentional descriptions of concepts [20] SKOS provides looser semantics than OWL [21].

The SKOS model can be used to structure and represent any knowledge that contains statements about concepts and the relationships between them. The shared features of these KOS are primarily in the form of a lexical resource along with some semantic relationships between each resource. The semantic relationships between resources are typified by broader, narrower, and related. SKOS provides a data model that can be used to express these kinds of relationships between resources and is designed to be extensible and modular. Central to SKOS is the core vocabulary deemed sufficient to represent most of the common features found in concept schemes. A *concept* can be considered any unit of cognitive thought. *Lexical labels* allow the association of lexical forms (preferred labels, alternative labels and so on) with each concept. *Semantic relations* capture relationships between concepts including hierarchical broader-narrower relationships and general associative relationships [22]. For example, a domain "*Biology*" is an individual of *Skos:Concept* in this ontology. We refer to individuals in this work by concepts. "*Biology*" may have subbed domains like "*Ecology*" and "*Botany*". Each of them is an individual of *Skos:Concept* too. They are related to "*Biology"* by narrower (more specific) and broader (more general) relations. We consider each concept as one federation.

TABLE I. THE SKOS ENCODING, IN TURTLE NOTATION [23], FOR THE ONTDD CONCEPT.

```
<#Ecology>
a skos:Concept;
skos:altLabel "Bionomics"@en, "Environmental science"@en;
skos:broader <#Biology>;
skos:definition "the branch of biology concerned with the relations
between organisms and their environment."@en;
skos:narrower <#Paleoecology>;
skos:prefLabel "Ecology"@en, "Ecologie"@fr;
```

We enhance and enrich the semantic meaning of concepts by the creation of different alternate label relations to represent its synonyms, and another property assertion called *associated_term,* which represents terms associated to the concept other than its synonyms.

A classification technique will classify each node according to its interest into a concept of OntDD. It will affect each node to the appropriate federation. We believe that if concepts are well defined, the use of a simple measure of similarity will be efficient and precise. Concepts in OntDD are already enriched by their semantic synonyms according to the context extracted from *WordNet* [24]. The *Levenshtein* measure will be used to calculate similarity between the domain of interest of node *N* wishes to join the grid and each federation *F* of OntDD. The similarity between *N* and *F* is the max of similarities between *N* and the set of all labels and associated terms of *F*. The node will be affected to the federation with the high similarity.

After a classification algorithm has been determined, the system can classify nodes and create federations. Figure 1 shows the system architecture.

B. Layred Architecture based Semantic Federation

As illustrated in Figure 1, we propose a hybrid layered architecture in which, each federation is structured following leader-workers paradigm to perform data retrieval of available resources.

From the bottom to the up, we have:

- The physical layer: represents nodes in a real network as unstructured network architecture. Edges in this layer represent physical connections.
- The federations' layer: represents the overlay network applied in this work to maintain federations and process queries. Each federation captures a concept defined in OntDD and has a leader and workers. A leader is a representative of its federation, which is selected among the other nodes; each leader node has links to all leader nodes, and links to all of its own worker nodes. Communication is limited on two sorts between leaders and between a leader and its workers, which reduce the overhead.
- The leaders' layer: since each concept in OntDD represents one node which is the leader of the federation, leaders can be organized as a hierarchical structure. This hierarchy between federations leaders is generated by traversing the configured properties *Skos:narrower* and *Skos:broader*, which are used to express the hierarchical relations between concepts in our case federations' leaders. In addition, with SKOS, we could organize federations (which are individuals of *Skos:Concept*) into categories using subclasses of *Skos:Concept*, we called them *MajorDomainFederation.* A category serves as grouping mechanism for concepts (leaders' federations) of the same inherent category, concepts being an instance of one of those categories. E.g.: "*MajorBiologyFedation"* is a category of all federations with their domain is one field of *Biology*.

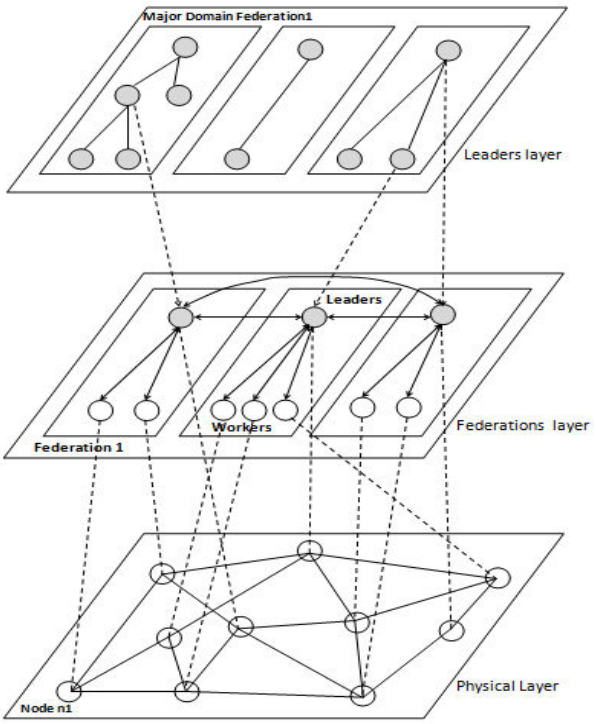

Figure 1. The hybrid layered architecture.

This hierarchy organization aids to limit the query search space from the entire grid to a federation through a single step and resource location inside the federation in the next step. If the federation is unable to respond to the query, it forwards the query to its relative hierarchy leaders that may satisfy the query. This forwarding mechanism between federation leaders achieves high resource discovery efficiency by keeping resource discovery scope at the federation leader level.

IV. ALGORITHMS FOR THE SYSTEM

In this section, we will present the algorithmic details of this system. We will discuss in detail how federations are maintained in terms of add and remove nodes. How a leader is elected and how queries are handled.

A. Nodes Joining and Leaving

When a node joins the network, it connects to any existing node in the network by sending a subscribe message, if this last node is not a leader node, it transfers the request to its leader. The leader calculates the similarity between the domain of interest of node and concepts of OntDD, then assigns it to the appropriate federation following the Table II.

TABLE II. THE ALGORITHM OF JOINING THE GRID.

Algorithm Join (*N*, *X*): Node *N* joins the Grid through node *X*

```
If X.isleader = true then
    Calculate the similarity between N and concepts of OntDD;
    Assign it to its appropriate federation.
  If N ∈ this-fedration then
    Update (addition) knowledge base of resources.
  Else
    If N ∈ other-federation then
      Send a message to its leader's federation.
    Else /* there is no adequate match*/
      N creates a new federation with itself as leader.
    End if
  End if
Else /* X is not a leader*/
    Transfer a subscribe message to the leader.
End if
```

To leave the grid, the node just sends a message of unsubscribing to its leader. If the leader wants to leave, a replacement of a leader occurs by selecting a new leader to preserve the federation's knowledge, and then the leader can unsubscribe.

B. The Process of Leader's Election

A good resource discovery mechanism based on leader-workers, should be able to select the best node to be a leader, to periodically check if the actual leader is the most pertinent, and to prevent leader failure to make the mechanism fault-tolerant. It should have the ability to detect the failure and to replace dynamically a failed leader. It exists several works on the leader election problem. [25] Proposed an election method where each node is assigned a unique steady ID and the node with the highest ID wins the election. The stability of ID even if the resources' node changes may render the current leader performances less than existing nodes. [26] Used a voting based system where each node casts a vote for which node it prefers for the leader role. In such a system, a mechanism for determining when the election begins and ends must be designed, since distributed election algorithms depend on a clearly defined exchange of information between nodes in order for each node to unanimously agree on the new leader [27, 28]. [29] Used a distributed mechanism of election where all nodes should agree on the future leader, it doesn't handle the case when a leader node was failed, and it uses a gossip messages to elect a leader and to inform all the other leader's groups about a new leader in order to update their information, this method is very expensive.

These mechanisms may generate a lot of traffic, particularly if the elections need to be restarted due to corrupted packets of intermittent network failures. A central leader election algorithm [30] is more striking to distributed leader election, since the leader choice is performed by a single node. In our approach, we will adopt this mechanism.

The election of leaders takes place periodically to check if the current leader is the most suitable. At the first time, the node that triggered the creation of the federation will be a leader of this federation. Later, each node in the federation can participate to this process; it has to calculate its proper reputation score. Every time the process is started nodes send their reputation score to the leader, this last selects the three nodes with the highest scores, the leading will be the new leader and the others will become the safeties nodes, then it informs the whole federation about them. Safety nodes act as workers, they are introduced to be as a secure in case where the actual leader was failed or want to leave. This mechanism avoids performance degradation and federation dissolution if a leader fails, because the whole system has already prepared its future leaders, which make the system fault tolerant. The leader sends copies of information to safety nodes every time it makes an update. Once the failure of leader is detected, a safety node with the high reputation score will be a new leader. The reputation score is calculated based on nodes characteristics such stability (it means that node doesn't leave or fail frequently) and the interne characteristics like CPU speed, RAM and HDD sizes, bandwidth.

To overcome the problem of communication overhead between leader's federations each time a new leader is elected. We give for each federation, independently on the real address of its leader, a virtual static address, using another asserted property into OntDD to assign to federations their virtual addresses. The leader then must associate its own address with the virtual one of its federation. The virtual address is stable, if a new leader is elected; all it has to do is to assign its proper address to the virtual one of the federations, without need to communicate and to publish it to the other leaders, for them, the address of the leader federation is not modified even if the leader was changed.

C. Semantic Query Processing Mechanism

The mechanism used in this work, uses OntDD to semantically propagate the query between semantic related federations. It decides where the query must be sent in the next step using the different semantic relationships seen in Section 3. This mechanism divides the space of query search on three spaces. It limits the query search space from the entire grid to these three spaces.

- Space 1: it represents the federation itself, the leader and its workers. In this space, the leader supports the search of resources in respond of the query in its knowledge base.
- Space 2: it represents federations inside the MajorDomainFederation. Where federations are related by hierarchical relationships narrower/broader and they are members of the same MajorDomainFederation. These federations are likely capable of responding on query, since they have close domains of interest as the leader who sent the query.
- Space 3: it represents federations related to the actual federation by the associative relationship *Skos:related*. In SKOS, an associative link between two concepts indicates that the two are inherently "related", but that one is not in any way more general than the other.eg. *Business* is related to *Statistics*. *Biology* is related to *Medicine* and *Business*. These federations are possibly capable of responding on the query because their domains of interest are related to the leader who sent the query.

TABLE III. ALGORITHM OF QUERY PROCESSING.

Algorithm Handle_Query (*Q*): *Q* is sent to leader *L* from node *n*

```
If n.isWorker = false then
   Find node(s) N in the federation that satisfy Q.
  If N ≠ {} then
    The search is succeeded; send a response to a requester.
  End if
Else /* n is one worker of L*/
   Find node(s) N in the federation that satisfy Q.
  If N ≠ φ then
    The search is succeeded; send a response to a requester .
  Else /* no node was found*/
    Forward the query, direct Q to leaders' in MajorDomainFederation
    Wait responses for a time T.
     If T=0 and no response then
       Forward the query, direct Q to the related leaders' .
       Wait responses for a time T'.
       If T'=0 and no response then
          The search is failed.
       Else /*one or more related leader could satisfy the query*/
          The search is succeeded; send a response to a requester.
       End if
     Else /*one or more leader from the MajorDomainFederation
          could satisfy the query */
          The search is succeeded; send a response to a requester .
     End if
  End if
End if
```

These three layers of search spaces achieve high resource discovery efficiency by keeping resource discovery scope at the federation layer and its related leaders, and will reduce the network traffic and the number of messages compared to the simple query flooding, or random-walk.

A query request is submitted to a leader node from one of its workers or another leader node. The leader follows two behaviors depending on the source of the request, with its workers it tries to find in its federation a worker node able to satisfy the query based on the leader's knowledge. If such a worker is not available, the leader sends the query to leaders' federations in its MajorDomainFederation, which they are likely to satisfy the query. If it doesn't receive any response after a while, it forwards the query to its related leaders' federations as the last resort. If a leader is solicited, it tries to find workers that respond to the request; otherwise, it ignores the query. This strategy of semantic query processing reduces the search time and decrease the network traffic by minimizing the number of messages circulating among nodes and federations. Table III, resumes this strategy.

V. PERFORMANCE EVALUATION

In this section, we present a theoretical study to evaluate the performance efficiency of our algorithm of query processing. With semantic federations' topology, resource discovery can be efficiently performed. In most cases, a resource can be located, within querying nodes with the same domain, and semantically related nodes that are within the neighborhood of the querying node in terms of related federations and MojorDomainFederation.

Supposing the number of grid nodes is N and the resource searched is r. We divide the whole grid according to our algorithm into M federations, K MajorDomainFederations, each MajorDomainFederation has P federations, L nodes per federation, Q related federations (if they exist) for each federation, and R messages as responses if they exist.

We evaluate the performance of our algorithm by comparisons with flooding based algorithm, [31], Random walk and [32]. We evaluate these algorithms by estimating the number of messages propagated in the network, and the number of hops needed to find a resource during one cycle of searching, and we discuss the theoretical efficiency of each one.

With flooding-based, node X that searches for a resource r checks its resource list, and if the resource is not found there, X contacts all its neighbors. In turn, X's neighbors check their resource lists and if the resource is not found locally, they propagate the search message to all their neighbors. The method ends when either the resource is found or a *TTL* is expired, in this case, the number of messages increases exponentially to the number of nodes. The number of hops is estimated as $T>>N$, thus it is not scalable.

With randomize walk strategy in pure P2P model, the number of nodes visited during the searching process is $\log(N)$, the number of hops is $O(\log(N))$, so a good performance is expected; However, this kind of algorithm is slow and no guarantee of actually finding the resource even if it exists, thus not efficient.

For [31], it uses a super-peer topology.

- Best case: one message with one hop. The resource is inside the requested cluster.
- Average case: 1+ O (log (P)) hops. As it uses a random walk for searching, the number of messages is logarithmic in the number of clusters P in the super cluster called Resource Classified Space, P corresponds to the number federations per MajorDomainFederation in our case.
- Worst case: 1+ O (log (K)) + O (log (P)) hops. In this stage, it uses a random walk and chord algorithms for searching, the number of messages is logarithmic in the number of clusters P in the super cluster, and the number of nodes K in the routing table of entry nodes, K corresponds to the number MajorDomainFederation in our case.

For [32], it uses a super-peer topology; it organizes nodes into groups with a leader-worker approach using KNN algorithm.

- Best case: one message with one hop. The resource is inside the requested group.
- Worst case: two hops with $1+M+R$ messages. It forwards the query to all the other groups in the grid. M is the number of groups in the grid; it corresponds to the number of federations in our approach.

Our strategy divides the space of search on three; this will conduct us to these situations:

- Best case: one message with one hop. The resource is inside the requested federation.
- Average case: two hops with $1+P+R$ messages. The resource is inside the MajorDomainFederation.

- Worst case: three hops with $1+P+Q+R$ messages. The resource is inside the related federations. This number represents the total number of messages generated during the whole process, it is the cumulative of the first, the second and the third case, in other words, this step generates just $Q+R$ messages at a time.

By comparing the estimated performance efficiency (number of messages and hops) of several algorithms, we could assume that our algorithm outperforms the flooding-based algorithm, randomize walk algorithm and algorithms presented in [31] and [32].

VI. Conclusion And Future Works

As more and more the scale of grid growing, there is a convincing need to find an effective and efficient way to organize nodes in order to facilitate the discovering and the querying of resources of these nodes. In this paper, we have presented a novel semantic approach of regrouping nodes into federations using SKOS ontology, to construct a three layered architecture. As shown; the propagation of query in this architecture is scalable and efficient since the space of querying is diminished from the entire grid to a smaller range consisting of three semantically related spaces in the worst case, which decreases the cost of resource searching. In addition, this architecture is helpful to enlarge the scale of grid. We have discussed the problem of leader election, and proposed an efficient process that rendered our system more scalable and fault tolerant.

However, this work is limited to theoretical discussion; the study to evaluate the performance of our algorithm in practice is our future work.

References

[1] A. Miles, and S. Bechhofer, "SKOS simple knowledge organization system reference". W3C, available at http://www.w3.org/TR/skos-reference. January 25 2008.

[2] D.L McGuinness, F.van Harmelen, "OWL Web Ontology Language overview". Recommendation, W3C, http://www.w3.org/TR/owl-features/. February 10. 2004.

[3] E. Meshkova, J. Riihiarvi, M. Petrova, and P. Mahonen, "A survey on resource discovery mechanisms, peer-to-peer and servicediscovery frameworks", Computer Networks 52, 2008, pp. 2097–2128.

[4] R. Raman, M. Livny, and M. Solomon," Matchmaking: Distributed Resource Management for high Throughput Computing", Proc Of the 7th IEEE HPDC, IEEE Computer Society Press, Washington DC, 1998, pp.140-146.

[5] A. Iamnitchi, and I. Foster, "On Fully Decentralized Resource Discovery in Grid Environments",Proc The 2nd IEEE/ACM International Workshop on Grid Computing, Denver, November 2001.

[6] H. Lican, W. Zhaohui, and P. Yunhe. "A scalable and effective architecture for Grid Services discovery", Proc of the 1st Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with the Twelfth International World Wide Web Conference,2003.

[7] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web". Scientific American 284(5),2001,pp.34–43.

[8] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. SIntek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "Edutella: A P2P networking infrastructure based on RD", Proc of the International World Wide Web Conference,WWW, Honolulu, Hawaii, USA, 2002, pp. 604_15.

[9] M. Arumugam, A. Sheth, and I.B. Arpinar, "Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web", Proc of the International World Wide Web Conference,WWW, Honolulu, Hawaii, USA, 2002.

[10] A. Halevy, Z. Ives, J. Madhavan, P. Mork, and D. Suciu, "The Piazza Peer Data Management System", 2004 pp.787-798.

[11] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M.T. Schlosser, I. Brunkhorst, and A. Lser, "Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks", Proc of International World Wide Web Conference,WWW, 2003, pp. 536-543.

[12] A. Crespo, and H. Garcia-Molina,"Semantic overlay networks for p2p systems",Technical report, Stanford University, 2002.

[13] A. Iamnitchi, M. Ripeanu, and I.T. Foster, "Locating data in (small-world) peer-to peer scientific collaborations", Proc of International Workshop on Peer-to-Peer Systems, IPTPS, 2002, pp. 232-241.

[14] X. Tempich, S. Staab, and A. Wranik, "REMINDIN': semantic query routing in peer to peer networks based on social metaphors", Proc of International World Wide Web Conference, WWW, New York, USA, 2004, pp. 640-649.

[15] S. Castano, A. Ferrara, Montanelli, D. Zucchelli, and Helios: "A general framework for ontology-based knowledge sharing and evolution in P2P systems", Proc of DEXA WEBS Workshop, IEEE, Prague, Czech Republic, 2003, pp.597-603.

[16] J. Li, "Grid resource discovery based on semantically linked virtual organizations", Future Generation Computer Systems 26, 2010, pp.361-373.

[17] http://code.google.com/p/skoseditor/

[18] https://sourceforge.net/projects/skosapi/

[19] http://protege.stanford.edu

[20] I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language". Journal of Web Semantics, 2003, pp.7–26.

[21] S. Bechhofer, Y. Yesilada, R. Stevens, S. Jupp, and B. Horan, "Using Ontologies and Vocabularies for Dynamic Linking". Internet Computing, 2008, pp. 32–39.

[22] S. Jupp, S. Bechhofer, and R. Stevens, "A Flexible API and Editor for SKOS", ESWC, Springer 2009, pp. 506–520.

[23] D. Beckett, T. Berners-Lee, "Turtle - Terse RDF triple language". Team submission available at http://www.w3.org/TeamSubmission/turtle/, w3c , 2008.

[24] http://wordnet.princton.edu/

[25] Garcia-Molina, " Elections in a Distributed Computing System". IEEE Trans. Computers, pp. 48–59, 1982.

[26] S. Singh and J. Kurose, "Electing leaders based upon performance: The delay model", 11th International Conference on Distributed Computing Systems, 1991, pp. 464–471.

[27] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, S. Spring, A. Su, and D. Zagorodnov, "Adaptive computing on the grid using apples", 2003.

[28] J. L. Kim and G. G. Belford, "A robust, distributed election protocol",Symposium on Reliable Distributed Systems, 1988, pp. 54–60.

[29] A. Padmanabhan, S. Ghosh, and S. Wang J, "A Self-Organized Grouping (SOG) Framework for Efficient Grid Resource Discovery", Grid Computing, Springer, 2009.

[30] T. W. Kim, E. H. Kim, J. K. Kim, and T. Y. Kim, "A Leader Election Algorithm in a Distributed Computing System", FTDCS, 1995, pp. 481–487.

[31] X. Wang, L.F. Kong, "Resource Clustering Based Decentralized Resource Discovery Scheme in Computing Grid", Proc of the 6th International Conference on Machine Learning and Cybernetics, IEEE, Hong Kong, August 2007, pp. 19-22.

[32] Y. Zhang, Y. Jia, X. Huang, B. Zhou, and J. Gu, "A grid Resource Discovery Method Based on Adaptive k-Nearesr Neighbors Clustering", COCOA, Springer, 2007, pp. 171-181.