A New Protocol for Resource Discovery in Grid Systems

Mahmoud El Samad
Lebanese International University (LIU)
School of Arts and Science
Computer Science Department

Amer Bakhach
Lebanese International University (LIU)
School of Arts and Science
Computer Science Department

Mohammad Hussein
Lebanese French University of Technology and Applied Sciences,
Deddeh, El Koura, Lebanon

ABSTRACT
Resource discovery is a real challenge in grid systems due to the dynamicity of nodes (i.e. any node can join or leave the system at any moment). This paper proposes a new protocol for resource discovery in dynamic grid systems. The hypothesis is that a grid is composed from a set of Virtual Organization (VO). The idea is to define a Distributed Hash Tables (DHTs) for each VO. The discovery inside a VO is a traditional discovery based on DHTs. The resource discovery between Virtual Organizations, i.e. between DHTs, is achieved through a new protocol enabling a persistent communication between all the VOs. The main advantage of the proposed protocol is to enable a robust global discovery between unstable VOs of a grid (any node or even VO can leave the system at any moment). We evaluate the proposed protocol by experiments showing its feasibility and benefits.

General Terms
Algorithms, Large-Scale Distributed Systems

Keywords
Resource Discovery, Grid Systems, Peer-to-Peer Systems.

1. INTRODUCTION
Nowadays, the grid technology allows the development of efficient services for sharing resources (e.g. CPU, I/O, data, services) between several Virtual Organizations VOs [1]. Resource discovery is a hard task in grid systems because these systems differ from traditional distributed systems by the instability and the large scale properties. The system instability (i.e. dynamicity of nodes) means that every node can join or leave the system at any moment.

In the literature, most of resource discovery methods are based on Web services [2, 3]. The first methods are implemented according to a centralized approach [4]. For example, the first version of the Monitoring and Discovery System (MDS-1) of Globus employed a centralized index including the information describing the resources (e.g data, CPU, I/O). New methods based on web services are implemented in a hierarchical topology. The idea is to associate to Every virtual Organization (VO), a web service to manage its resources. The interconnection between VOs is made according to a hierarchical approach which allows a better scale than the centralized approach. The main advantage of these methods is the compatibility with Open Grid Service Architecture (OGSA). Nevertheless, the hierarchical approach is badly adapted to the system instability.

Peer-to-Peer (P2P) techniques were adopted to resource discovery in grid systems [5] due to their "good" characteristics such as decentralized control, scale and dynamicity. Current methods, based on P2P techniques, for resource discovery in grid systems can be classified into three classes [5] depending on the P2P architecture: unstructured [6, 7], structured [8,9,10] and super-peer methods [11,12].

In the P2P unstructured methods, nodes often communicate by diffusion which could create a flooding in the system. Hence, these methods are not scalable. Structured P2P methods allow a better scale [5,10] by using Distributed Hash Tables DHTs. DHTs allow distributed resource discovery in $O(Log(N))$ messages where N is the number of nodes in the system [13,14]. The DHTs must be updated when a node joins or leaves the system which costs $O(Log^2(N))$ messages [13]. The use of a single DHT for resource discovery in all the system [15] does not take into account the existence of many VOs (in the system) where every VO is dedicated to a domain (e.g. finance). This leads into two inconveniences. The first inconvenience is that the message traffic ($O(Log^2(N))$ messages) which is generated by the dynamicity of nodes, might be very dense if the number of nodes is relatively high. The second inconvenience is that the use of a single DHT does not take into account the principle of locality [16] during the resource discovery phase. Generally, users often access resources in their domain, i.e. in their VO. So, it is important to launch the resource discover in the local VO (i.e. the VO of the user who submits a query) rather than in all the system.

In the super-peer methods [11,12], resources are first discovered in the local VO. Then, the discovery is propagated to all other VOs in the system or to a set of neighbor VOs. A super-peer manages the resource belonging to its VO in a centralized or hierarchical fashion [12]. The super-peers can be organized in an unstructured peer-to-peer layer. The inconveniences of these methods are: (i) the creation of a bottleneck if a large number of messages are sent to the super-peer and (ii) the failure of the super-peer results the VO to become unusable and inaccessible by other VOs.

A simple idea to avoid the problems of super-peers and the use of a single DHT consists in defining a DHT for each VO in the system. Each DHT acts as distributed index for managing the resources for a given VO. First, this solution applies the principle of locality [16] if the resource discovery query is first submitted in the local VO. Second, using a DHT per VO allows a better scale than using a super-peer per VO. But this solution still has a problem which is: how to send a message for resource discovery towards other VOs if the discovery in the local VO fails without relying on a central node as in super-peers. Put differently, how to interconnect efficiently the VOs in the presence of dynamicity of nodes? In this paper, we propose a new protocol for resource discovery in grid systems. The idea is to define a DHT for each VO in the system. A $(DHT)_i$ acts as distributed index inside a $(VO)_i$. Two levels are defined for the resource discovery: local discovery and global discovery. The local discovery is first submitted in the local VO (principle of locality) with a traditional resource discovery based on DHTs. If the local

discovery fails, i.e. if the resource do not belong to the local VO, the global discovery is propagated to a subset of VOs (e.g. in case we need to discover computational resource) or to all other VOs (in case we need to locate a resource of type data). The communication between the VOs is made using a simple but original protocol [10]. The protocol allows reaching any VO without relying on a central node in a VO. More precisely, the protocol allows for each node *Ni* of a VO_i to own the address of a node N_j for each VO_j. Thus, N_i could reach any VO_j in the system. The nodes (N_j)s are different in a VO_j in order to avoid problems related to central node (i.e. the bottleneck and the fail of the central node). The main advantage of our protocol is to allow a robust global discovery in an unstable system. The protocol does not rely on any central node(s) in the system.

The rest of the paper is organized as follows: section 2 describes the proposed method for an efficient resource discovery in grid systems. Section 3 describes the protocol while section 4 describes the performance evaluation. Finally, section 6 concludes and presents the perspectives.

2. A NEW PROTOCOL FOR RESOURCE DISCOVERY

Suppose that a grid is composed of a set of VOs [1]. When a query is submitted, the resource discovery is processed first in the local VO (principle of locality). In case of failure (i.e. the desire resource is not found in the local VO), the discovery is propagated to all other VOs or a subset of VOs. Our work is limited to the discovery of resources using simple keywords [10] (e..g discover a file, a database). We do not handle complex queries such as multi-attribute and range queries [17]

2.1 Local discovery

The local discovery is a traditional discovery, based on DHT. A DHT allows the storage and the access of pairs of type <*key*,*value*> and associates for every *key*, a *value*. The access to a *key* is made in *O(Log(N))* hops where *N* is the number of nodes in the system. Hence, the DHT provides primitives such as: *lookup(key)* (returns the *value* associated with the *key*) and *store(key*, *value)* (associates a for *key*, a *value*). In our system, the resource discovery is made with the primitive *lookup(R,VOloc)* where *R* is the name of the resource and *VOloc* corresponds to the local VO. If *R* is stored in the local VO than the system reply by sending the description of *R*: *InfoR* which contains the description of *R* (e.g. its location, size,…etc).

2.2 Global discovery

If the local discovery of *R* fails then the discovery should be propagated towards all VOs belonging to the system. This action is called a global discovery. The main problem that we tackle in this paper is how to propagate the discovery of a resource *R* to all other dynamic VOs in the system. We propose a new protocol which enables a robust communication between VOs. The idea is that each node N_i of a VO_i own the address of a node N_j for each VO_j belonging to the grid. A node *Nj* will be named the connection point of *Ni* to VO_j. The nodes (*Nj*)s are selected differently in order to avoid the inconveniences of a central node (i.e the bottleneck and the fail of the central node). The nodes of the same VO_i owns different connection points to any VO_j in the system. A node N_i of a VO_i knows a single connection point for each VO_j. Hence, the number of connection points for a node is equal to the number of VOs in the system minus one.

The last contacted node *currentNode* in the local DHT (during the local discovery) notices the failure of the discovery in the local VO; therefore, *currentNode* initialize the process of the global discovery by propagating messages towards all other VOs using its connection points. If *currentNode* contacts an connection point *apDisconnected* which does not react during a certain period of time (e.g. a Round-Trip Time RTT) then *currentNode* contacts its neighbor via its local hash table to get another connection point to the desired VO. In the rest of the paper, the term neighbor will be used to indicate the next neighbor (which is the closest neighbor) in the DHT. This is repeated recursively by asking the neighbor of my neighbor until finding a connected connection point *apConnected*. When *currentNode* finds an *apConnected* then it must update its *apDisconnected*. So *currentNode* asks the *apConnected* to send the address of its neighbor. This strategy will guarantee that the connection points towards a VO are not the same. This will avoid a bottleneck (in case of strong instability) on a single point.

The connection points (owned by neighbors) that are contacted during the global discovery but did not reply (because probably they are disconnected) will not be updated, i.e. they will not be replaced by another connected connection points. In fact, this update can create a very high maintenance cost if a large number of nodes leave the system or if a VO is disconnected. Therefore a lazy update [10] is adopted which consists in updating only the connection point owned by the node which submits the process of the global discovery. Hence the global discovery is robust. It works with presence of dynamicity of nodes and it allows distributing the load between all the nodes in a VO. The resource discovery algorithm of the protocol is described in Figure 1.

```
discoverResource(input:Resource R,
   output:information describing R){
  //R : a resource (e.g. a file, a relation)
  //cpi : connection point toward a VOi
  //locVO : local VO
  //disVOi : distant VOi
  infoR = currentnode.lookup(R,locVO);
  if <infoR is find in locVO>  {
     Send infoR;
   }else{
    for each cpi ∈ currentNode {
     if <cpi is not connected> {
      // This is done recursilvely by asking
      //my neighbor, then the neighbor of my
      //neighbor and so on :
      ask my neighbor a new cpk ∈ VOi;
      //updating the current cpi
      cpi ← neighbor (cpk) ;
     }//end if
     infoR ← cpi.lookup(R,disVOi);
     if <infoR is find in disVOi> {
       return infoR;
     }//end if
    }//end for
  }//end if
  return "Discovery of R fails" ;
}
```

Fig 1: The resource discovery algorithm of the protocol

2.2.1 Example of global discovery:

To illustrate the mechanism of the global discovery, suppose that we have a grid composed of 3 VOs: *VO1*, *VO2* and *VO3* (Figure 2). Each DHT is responsible of managing the resources for a VO. For the clarity of the figure, we show only the connection points of the nodes of *VO1* towards *VO2* and *VO3*.

Suppose that we have a user *U* from the *VO1*. *U* needs to access a resource (e.g. a relation *R)* that belongs to *VO2*. The process of local discovery begins, for example, from *N23*. The discovery of *R* fails in the *(DHT)1*. *N15* notices that the *R* does not belong to the *VO1*. In this type of scenario, a message is propagate for enabling global discovery. Thus, *N15* sends a message to the other VOs using the connection points: *N7* for the *VO2* and *N5* for the *VO3*. The resource discovery of *R* is submitted in parallel in the *VO2* and the *VO3*. The *VO2* communicates *infoR* via the node responsible of *R* while the discovery in the *VO3* fails.

In Figure 2, the connection points of *N15*: *N7* and *N5* are supposed to be connected. Suppose now that one of the connection points (e.g. *N7*) is disconnected (Figure 3). In that case, *N15* contacts its neighbor in its local hash table (i.e. *N20*) to get another connection point toward *VO2*. *N20* communicates *N19* which is connected. Now, *N15* can communicate with the *VO2* via *N19*. First, it asks *N19* the address of its neighbor (which is *N4*) in order to update its connection point toward *VO2*. Hence, *N15* replaces *N7* by *N4*. Second, the resource discovery of *R* is submitted in the $(DHT)_2$.

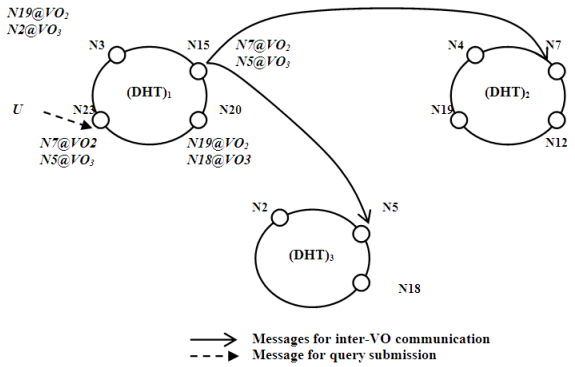

Fig 2: Global discovery absence of unstable nodes

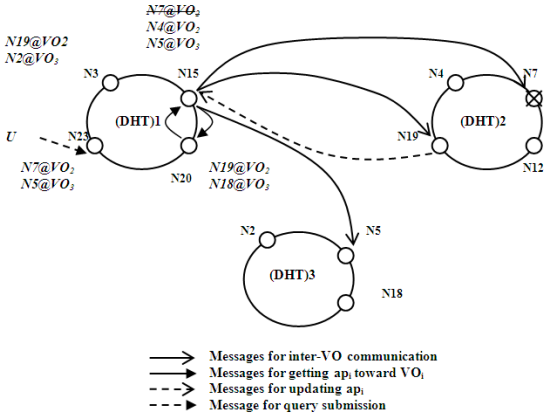

Fig 3: Global discovery with presence of unstable nodes

3. MAINTENANCE OF THE PROTOCOL

Due to the dynamicity of nodes, our system requires maintenance at two levels: (i) the maintenance of DHT_i (defined for each VO_i) and (ii) the maintenance of the protocol. The maintenance of DHT_i in a VO_i is a traditional maintenance [13, 14]. The maintenance of the protocol consists in defining how the connection points are established (resp. updated) when a node is connected (resp. when a node is disconnected).

3.1 Node connection

When a new node *Nnew* connects to a local VO (*locVO*), the local DHT is updated as in traditional DHT. Then, *Nnew* contacts its neighbor via its local hash table to get the connection points toward all VOs in the system. *Nnew* sends a message to every connection point *api* for each *VOi* in order to ask the ap_i the address of its neighbor. If an ap_i does not reply during a certain period of time (e.g. a RTT), then the ap_i is considered as disconnected. In that case, *Nnew* contacts the neighbor of its neighbor in the *locVO* to get another *cpi* toward *VOi*. This is repeated by a recursive call for the neighbor of neighbor and so on, until finding a connected connection point toward *VOi*. In case where all nodes of the *locVO* are contacted, the VO_i is considered as disconnected. In that case only, the information about the disconnection of *VOi* is broadcasted to all nodes of *locVO*. To simplify the algorithm, we do not present this case in Figure 4.
As we already mentioned in Section 2.2, the connection points that are contacted but did not reply will not be updated because the protocol adopt a lazy synchronization.

```
establishConnectionPoints(input: Nnew,
       output: set of connection points){
 //cpi : connection point toward VOi
 get all the cpi by asking my neighbor;
 for each cpi{
  if <cpi is not connected>{
   // This is done recursilvely by asking
   // the neighbor of my neighbor and so on
   ask the neighbor of my neighbor for a new cpi;
   //value exchange between the disconnected cpi
   //and the connected cpk
   cpi ← cpk ;
  }//end if
  //Assign a cpi to the new node
  cpi ← neighbor(cpk) ;
 }//end for
}
```

Fig4 : Algorithm for the establishment of connection points for a new node

3.2 Node disconnection

When a node *Ndisc* is disconnected from its local VO, a traditional update is done for the local DHT [21]. Then, we have two possibilities for the maintenance of the protocol: either *Ndisc* propagates the information towards all the VOs of the system in order to update nodes using *Ndisc* as an connection point, or *Ndisc* does not indicate its departure.

In the first case, *Ndisc* propagates the information towards every *VOi* to indicate its departure via its connection points *cpi*. Then, every *VOi* has to inform all its nodes of *Ndisc* departure. A flooding will be generated in every *VOi* in the system. This leads to very bad performances.

In the second case, no message is sent to indicate the departure of *Ndisc*. This strategy is more preferable because the resource discovery algorithm takes into account the presence of dynamicity of nodes (see Figure 1). In addition, the protocol adopts a lazy update which avoid a high maintenance cost (see Section 2.2). The connection points (towards VOs) for a given node N are updated, only if this node N needs to reach these VOs.

towards the *VO3*). *Nnew* contacts *N19* (resp. *N2*) to ask an address of its next neighbor. Finally, *N19* (resp. *N2*)

3.2.1 Example for the establishment of connection points for a new node:

Figure 5 illustrates the connection of a new node *Nnew* to the *VO1*. A traditional update of the local DHT is made [21] in the *VO1*. Then, it is necessary to establish the connection points for *Nnew*. So, *Nnew* contacts its neighbor, i.e. *N23*. *N23* communicates *N19* as a connection point towards the VO_2 (resp. communicate *N2* as a connection point towards the *VO3*). *Nnew* contacts *N19* (resp. *N2*) to ask an address of its next neighbor. Finally, *N19* (resp. *N2*) communicates *N3* (resp. *N5*) to *Nnew* via its local hash table.

In the example of Figure 5, the connection points (i.e. *N19* and *N2*) were supposed to be connected. Let us suppose now that a connection point is disconnected (e.g. *N19*) (Figure 6). In this case, *Nnew* contacts the neighbor of its neighbor, i.e. *N3* which communicate *N7*. Then, *Nnew* contacts *N7* (which is connected in Figure 6) to get the address of its neighbor, i.e. *N12*. Finally, *Nnew* memorize *N12* as a connection point toward VO_2.

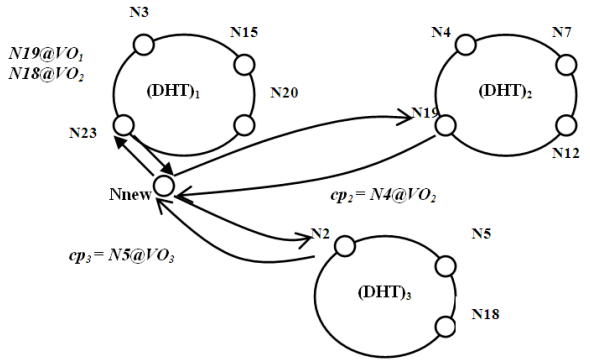

Fig 5: Establishment of connection points for *Nnew* in absence of unstable node

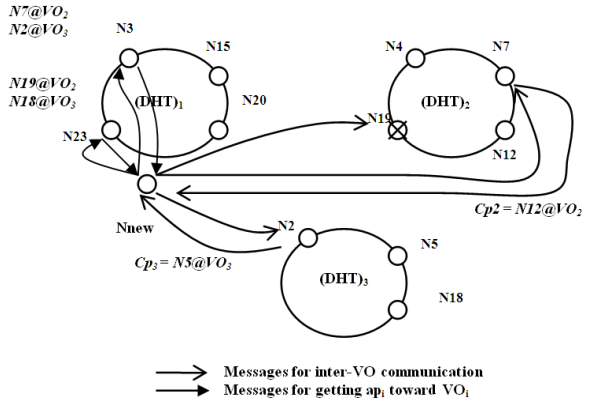

Fig 6: Establishment of connection points for *Nnew* in presence of unstable node

4. PERFORMANCE EVALUATION

This section presents the performance evaluation of our proposal using multiple DHTs (the MDHT method) for resource discovery with regard to: (i) the super-peer method [16] (named the SP method) and (ii) the structured P2P method based on the use of a single DHT for all the system [13,14] (named the SDHT method). We consider the same resources in the three methods. A resource discovery message consists of finding a resource based on keyword search (e.g. locate a database DB, locate a service SV)

The comparison is made according to two parameters: (P1) the message load [12] (i.e. the number of messages per second) for resource discovery and (P2) the number of nodes which connect (resp. disconnect) to the system (resp. from the system).

We compare our method MDHT only with regard to the SP method and the SDHT method. A comparison (by simulation) already exists in [12] between the SP method, unstructured P2P methods and the hierarchical methods. This comparison shows in many scenarios that the SP method is more convenient then the above methods.

4.1 Objectives

The objective of the performance evaluation is to study the impact of the parameter P1 on the average response time [12] to discover a relation with the MDHT method and the SP method (resp. SDHT the method). Then, we study the impact of the parameter P2 on the number of generated messages with the MDHT method and the SDHT method. The impact of the parameter P2 is not studied in the SP method because if a super-peer fails, the VO where the VO is responsible becomes unusable and inaccessible by the system.

4.2 Experiments

The behavior of the three methods (SP, SDHT and MDHT) must be studied on a large number (thousands) of nodes. The proposed method is implemented by using Past [18], a peer-to-peer system for storing the description of the resources in the DHT. Past enable us to query over Pastry [14]. We use FreePastry [19] an implementation of Pastry at Rice University in Houston. As the proposed method has been deployed on a single-processor machine, we limit ourselves to the creation of 5 VOs, each consisting of 100 virtual Pastry nodes. The routing module is simulated by a text file that maps for each $(VO)_i$ a hash function h_i. Finally, we tested the insertion and the discovery of resources in the $(VO)_i$ created to show the feasibility of our proposal. We will use parameters used in large-scale systems. Our platform allows publishing new resources (i.e. databases, services, files) periodically. Nodes can join and leave the system at any moment. To fairly compare the three methods, the same resource distribution is employed in all the experiments.

In the first part of the experiments (study of P1), the response time to discover a resource includes the communication cost and the local processing cost for the three methods [12]. In the SP method, the communication cost is equal to the hop cost between a peer and its super-peer if the local discovery succeeds. Otherwise, the communication cost is the sum of the hop cost between a peer and its super-peer plus the hop cost between two super-peers. The communication cost for the SDHT method is estimated to *O(Log(N))* hops [14] with *N* the number of nodes in the system.

In the second part (study of P2), we calculate the number of generated messages during the connection or the disconnection of nodes. In a DHT, the connection or the disconnection of a node generates $O(Log^2(N))$ messages with N the number of nodes in the system [14]. In the MDHT method, the connection of a node also generates $O(Log^2(N))$ messages (with N the number of nodes in a VO) plus the number of messages sent to contact connection points towards all others VOs in order to maintain the protocol. While the disconnection of a node generates $O(Log^2(N))$ messages because the protocol is employing a lazy synchronization.

In the three methods (SP, SDHT and MDHT), the hop cost between any two nodes is set to 30 ms and the local processing cost is set to 0.2 ms [20].

4.3 Impact of the message load

In this part, the message load of the resource discovery is incremented. These messages are sent by random nodes according to two scenarios: the first one supposes that the relation to discover belongs to the local VO (i.e. local discovery). The second supposes that the relation belongs to another VO (i.e. global discovery). These scenarios are valuable for the SP method and the MDHT method. While for the SDHT method, there is not distinction between a local or a global discovery because the DHT is constructed for the whole system. In the SP and MDHT methods, if the local discovery fails then the discovery is propagate to all other VOs in order to have a fair comparison.

4.3.1 The SP method VS the MDHT method

Figure 7 (resp. Figure 8) presents the average response time for the local discovery (resp. global discovery) according to the message load with the SP method and the MDHT method. Behaviors were different in the:

- SP case: when the message load increases, the average response time to discover a resource increases in a significant way.
- MDHT case: when the message load increases, the average response time to discover a resource remains almost constant.

The SP method is better than the MDHT method for a message load around 100 messages per second during a local discovery (resp. 200 messages per second during a global discovery). Indeed, the messages for the resource discovery of relations are executed sequentially by a super-peer in the SP method which increases the average response time (the local processing and the communication costs). On the other hand, the messages for the resource discovery are executed in a parallel way in the MDHT method which allows having an almost constant average response time.

The speedup is about 10 for 1000 messages per second during a local (resp. 5 for a global discovery). The speedup is the ratio between the average response time with the SP method over the average response time with the MDHT method. In a large scale environment, the message load can increase dramatically and hence the MDHT is better than the SP in this case.

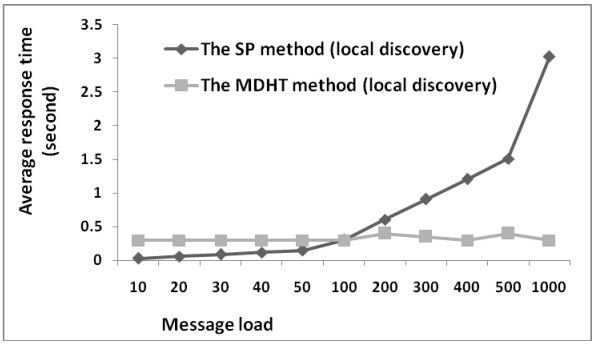

Fig 7: Impact of the message load (local discovery)

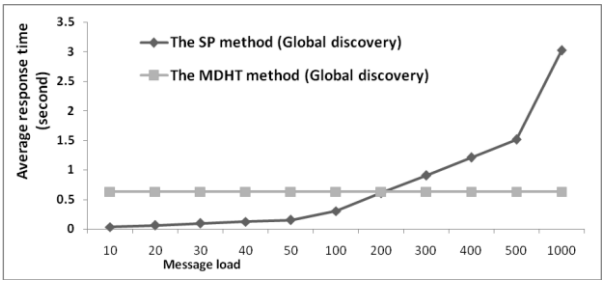

Fig 8: Impact of the message load (global discovery)

4.3.2 The SDHT method VS the MDHT method

Figure 9 presents the average response time to discover a resource according to the message load with the SDHT method and for the MDHT method.

In the case of a local discovery, the MDHT method is 15% better than the SDHT method. This result is expected because the number of nodes in a VO is smaller than the number of nodes in the system.

In the case of a global discovery, the SDHT method is 20 % better than the MDHT method. Indeed, in the MDHT method if the discovery fails in the local DHT then the discovery is propagated (in parallel) towards all other DHTs which provokes an additional cost.

Finally, we notice for the 3 curves that the average response time is almost constant expect the global discovery in the MDHT method. Indeed, global discovery requires contacting connection points towards distant VOs which can be possibly disconnected so it needs to contact other ones. This provokes an additional cost which is acceptable.

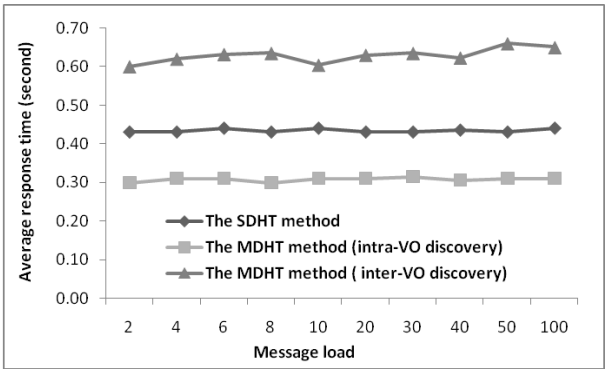

Fig 9: Impact of the message load

4.4 Impact of the number of nodes which connect or disconnect

According to Figure 10, The MDHT method is always better than the SDHT method. Indeed, the protocol does not require a high maintenance cost. In the case of the connection or the disconnection of 1 node (resp. 20 nodes), the SDHT method generates 32.4% (resp. 50.7%) messages more than the MDHT method. The speedup of the MDHT method with regard to the SDHT method is goes up to 2.04 (for 20 nodes).

In the MDHT method, the connection of a node generates more messages than the disconnection of a node. Indeed, the connection of a new node *Nnew* requires an additional number of messages in order to establish the connection points from *Nnew* towards all VOs in the system. While the disconnection of a node is not indicated to the protocol (see section 3.2).

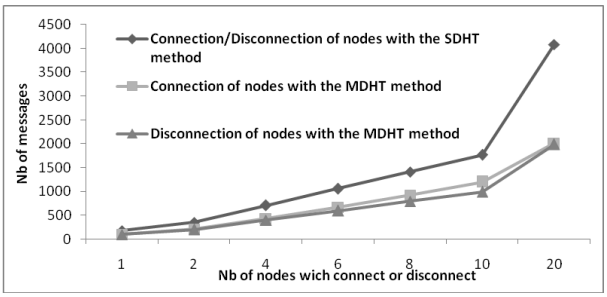

Fig 10: Impact of the number of nodes which connect or disconnect

5. CONCLUSIONS AND PERSPECTIVES

This paper proposed a new protocol for resource discovery enabling a robust communication between DHTs where each DHT is associated to a VO. Our solution can operate in a large scale environment with respect to system instability (any node or VO can join/leave the system). The work was focused on the resource discovery based on simple keywords (e.g. locate a database, locate a service).

The performance evaluation shows that our proposal is better than the super-peer method [16] for a message load around 100 messages per second in local discover (resp. 200 messages per second for global discovery). Moreover, our proposal is better than methods based on a single DHT [7, 19, 21] when the users submit query in their VO. Otherwise, the methods based on a single DHT are better. Finally, we notice that unpacking a single DHT to many DHTs reduces significantly the maintenance cost of the DHTs. The proposed protocol is maintained automatically during the global resource discovery. A lazy synchronization is adopted for the protocol to reduce the maintenance cost due to node departures.

In the near future, we will focus on: (i) the study of the behavior of our method while modeling the behavior of node arrivals/departures [21] in a more detailed way and (ii) the extension of our method to computational resource discovery including complex queries [17].

6. REFERENCES

[1] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, Elsevier, 2003.

[2] Huang et al., "Dart: A Framework for Grid-Based Database Resource Access and Discovery", LNCS, Vol. 3033/2004, Book Grid and Cooperative Computing, Springer 2004.

[3] A. S. Lynden et al., "The design and implementation of OGSA-DQP: A service-based distributed query processor", Future Generation Computer Systems, Vol. 25, Issue 3, pp. 224-236, 2009.

[4] Nirmala S Devi and A Pethalakshmi. Application of ACO for Resource Discovery in Grid Computing Environment. International Journal of Computer Applications 43(2):13-16, April 2012.

[5] P. Trunfio, et al., "Peer-to-Peer resource discovery in Grids: Models and systems", Future Generation Computer Systems, pp. 864-878, 2007.

[6] A. Iamnitchi and I. Foster, "A peer-to-peer approach to resource location in Grid environments", Grid resource management: state of the art and future trends, pp: 413 – 429, 2004

[7] E. Jeanvoine and C. Morin, "RW-OGS: An optimized random walk protocol for resource discovery in large scale dynamic Grids", Grid Computing Conference, IEEE/ACM, pp. 168-175, 2008

[8] H. A. Ali, "A Framework for Scalable Autonomous P2P Resource Discovery for the Grid Implementation", International Journal of Computer Science and Engineering, PWASET, Vol.25, 2007.

[9] Filho et al., "PerDiS: a scalable resource discovery service for the ISAM pervasive environment", International Workshop on Hot Topics in Peer-to-Peer Systems, pp. 80- 85, Oct. 2004.

[10] M. El Samad, "Resource discovery and monitoring in data grids", PhD thesis 2009, Paul Sabatier University.

[11] C. Mastroianni et al., "A super-peer model for resource discovery services in large-scale Grids", Future Generation Computer Systems, Vol. 21, pp. 1235-1248, Elsevier Science, 2005.

[12] C. Mastroianni et al., "Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models", Parallel Computing, Vol.34, issue 10, pp. 593-611, 2008.

[13] I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", ACM SIGCOMM Conference, pp. 149-161, 2001.

[14] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", Int.Conf. on Distributed Systems Platforms, pp. 329-350, 2001.

[15] D. Doval and D. O'Mahony, "Overlay networks: A scalable alternative for P2P", Internet Computing, IEEE, Vol. 7, pp.79-82, 2003.

[16] P. Watson, "Databases in grid applications: Locality and distribution", British national conference on databases, LNCS, No. 22, Vol. 3567, pp. 1-16, 2005.

[17] Torkestani, Javad Akbari, «A multi-attribute resource discovery algorithm for peer-to-peer grids", Applied Artificial Intelligence, Aug. 2013, Vol. 27 Issue 7, p575-598.

[18] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility", HotOS VIII, Schoss Elmau, Germany, May 2001.

[19] FreePastry. http://freepastry.rice.edu/

[20] P. Hasselmeyer, "The nextgrid project: architecture for next generation grid", work package 5, grid dynamics, document p.5.2.1, Tech. Rep., 2005.

[21] Wu et al.," Analytical Study on Improving DHT Lookup Performance under Churn", Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, pp. 249 – 258, IEEE 2006.