A New Grid Resource Discovery Framework

Mahamat Hassan and Azween Abdullah
Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia

Abstract: *Resource Discovery is an important key issue in grid systems since resource reservation and task scheduling are based on it. This paper proposes a novel semantic-based scalable decentralized grid RD framework. The paper integrates ontology, Peer-to-Peer network and intelligent agents to build the framework. The framework consists of an ontology model, an agent model, and a set of algorithms for implementing the P2P architecture and searching the shared resources. The paper shows how the framework satisfies grid RD features such as scalability, decentralization, dynamism and interoperability.*

Keywords: *Grid computing, peer-to-peer networks, ontology, and intelligent agent.*

Received May 7, 2009; accepted August 4, 2009

1. Introduction

Grid is a collection of shared, geographically distributed hardware and software resources made available to a group of remote users [5]. Resources in a grid can be a CPU, electronic devices, network, software application components, and so on. All these resources are connected via the internet.

Grid RD refers to the process of locating suitable resources based on users' requests [11]. This process represents an important step based on which resource reservation and task scheduling can take place to enable grid applications development. However, grids are associated with some complexities such as, grid resources (e.g., CPU, network and storage) are heterogeneous, dynamic and they tend to have faults that may not be predictable, and grids are often distributed across security domains with large number of varied resources [17]. These complexities have raised several requirements that should be addressed by any developed RD system. The requirements include decentralization, scalability, dynamism, and interoperability. Accordingly, a grid RD system should be fully decentralized from any global control, tolerates intermittent resource participation (either voluntary or due to failure) [16] and supports semantic description for resources and applications [10]. As a result, it is challenging indeed to develop efficient RD methods to discover the resources and fulfill the above mentioned requirements.

Currently, there are two types of grid RD systems, which can be classified into centralized and hierarchical systems. In centralized RD systems, the information on resources (metadata) is indexed under a centralized node, and users send their resource queries to that node. The resource providers update their resource status at periodic intervals using resource update messages. Condor system [2] is an example of the centralized systems. In Condor model, the centralized node is called Central Manager (CM), which collects information about the state of resources from resource providers. The resource providers are represented by Resource-owner Agent (RA), which is located in each resource provider. The CM then, receives users' tasks and matches them with the resources. In hierarchical systems however, the information on resources is indexed under a set of nodes in a hierarchical manner (each chilled node indexes its metadata on its parent node). The Monitoring and Discovery Service (MDS) of Globus [7] implements this model. It uses two services: a configurable information provider called Grid Resource Information Service (GRIS) and a configurable aggregate directory service called Grid Index Information Service (GIIS). A GRIS answers queries about the resources of a particular node. A GIIS combines the information provided by a set of GRIS services managed by a given Virtual Organization (VO).

Both of the systems (centralized & hierarchical) have some issues with regard to the RD requirements [10, 25]. For example, in Condor system, the Central Manager that matches the resources with the users' tasks may be a point of the failure. In Globus MDS, the updates on GRISs at the lowest levels do not automatically propagate up to the top of the hierarchy, which means the available resource information may not be completely up-to-date. This has motivated researches recently to focus on peer-to-peer based models for Grid RD systems [1, 3, 18] to ensure decentralization. However, they do not support semantic description/interoperability. This paper introduces a new grid RD framework that supports interoperability and fulfills other grid RD requirements/characteristics such as scalability, decentralization, and dynamism. To achieve these

goals, we integrate three technologies into the framework. These technologies are ontology, P2P network, and Intelligent Agents (IA). Ontology enables semantic communication in a domain, which is a means for interoperability [4, 12, 25]. P2P network allows resource sharing in a decentralized and dynamic manner, which offers system scalability [23]. IA has some useful characteristics to deal with complex and dynamic environments and acts on behalf of humans [20].

The rest of this paper is as follows; section 2 describes the contributions of our work, section 3 discusses some related technologies. The new RD framework is introduced in section 4. Section 5 presents an application. Section 6 discusses pertinent issues and section 7 concludes the paper.

2. Contribution

Our work aims at contributing to the development of sophisticated grid RD system that takes into account the identified requirements and at providing a scientific progress further than the state-of-the-art in this field. In contrast to most other works, we introduce ontology as a description mechanism for the grid resources and applications, P2P architecture to organize the metadata sources and intelligent agent to deal with dynamism of the metadata. Our framework provides an ontology model that can satisfy interoperability; a class-based node organization in which the nodes of resources are semantically grouped to allow them to index their resource information in a decentralized and scalable manner; and an agent model that shows how the nodes can cooperatively work during the resource discovery process. In short, with this framework, we are solving the shortcomings of both the current grid RD systems and the research oriented P2P based grid RD systems.

3. Related Technology and Study

This section discusses the related technologies that are used to build the new RD framework. These technologies are the JXTA P2P network, ontology and intelligent agents.

3.1. The JXTA P2P Network

JXTA is an open-source project originally created by Sun Microsystems [24] to offer a variety of services over the virtual overlay network (P2P network) [13]. JXTA has some basic components such as advertisement, peers, and peer group. Advertisement is a XML document that can describe resource information. Peer can act as the part of resource provider, resource consumer or hub/ super peer. When a peer is a hub, it is supposed to store the advertisements of the normal peers. Peer group is a set of related peers with their hub [28]. The search for resource in JXTA is achieved by distributing queries across a network of peers through hubs [15].

In this paper, we use the JXTA architecture to organize the grid nodes. In this case, nodes are classified into classes based on some predefined criteria, which is similar to JXTA peer group. Each class has a head that is elected among its own class nodes/members, which resembles the JXTA hub. The reason behind using JXTA architecture is to provide decentralization and scalability features in the grid environment. These features are available in JXTA P2P infrastructure and are yet to be implemented in the current grid models.

3.2. Ontology

Ontology is defined as the formal specification of a vocabulary of concepts and axioms relating to them [4, 12]. It formally specifies how to represent objects, concepts and other entities that are assumed to exist in some area of interest and the relationships among them [25]. Establishing relationships between domain concepts allows us to understand the concept not merely by its properties, but by its presence in relation to other concepts within the ontology [6].

In this paper, ontology is introduced for resource description and discovery, which may ease the communication between resource providers and consumers. There will be no ambiguity between resource provider and requester regardless of the middleware difference. It should be noted that, we are not creating a grid ontology, rather we model some criteria upon which an existing grid ontology such as the work of [19, 27] can be used here. Our model consists of several definitions as follows:

Definition 1: Ontology (O) consists of three entities: set of Concepts (C), Properties (P), and Relationship between those concepts (R). $O = \{C, P, R\}$, where, C is the set of grid resource/application concepts, P is the set concept properties, and R is the relations between the resource concepts, which can produce the concept hierarchy. For instance, a computer and operating system can be concepts, and the relation between these concepts is that computer *has an* operating system, and the property of the concept operating system is its version (windows, Linux, apple).

Definition 2: ontology must have completeness and expressiven*ess*. Completeness is that all the concepts hierarchy of the grid resources and applications is covered. Expressiveness means that the terms used in the ontology should be common to the participants.

Definition 3: ontologies are distributed; each grid node[1] has its local ontology as shown in figure 1. This allows a grid user from any node to describe/request

[1] We urge a grid node to be an actual physical organization due to security issues and organization policies.

resources based on the ontology. The Web Ontology Language (OWL) [14] can be used as the markup language for the ontologies. An obvious question here is how to manipulate the semantic information. Manipulation means the computation of the similarity between concepts, so that we can ensure the satisfaction of resource requests with respect to the described resource information. This computation can be done by a function as defined below.

Definition 4: a similarity function is a real valued function that computes the similarity degree between two concepts based on their properties. $Sim\ (x , y) : C \times C \rightarrow [\ 0 - 1\ ]$, where x and y are concepts, the value $sim\ (x , y)$ ranges between 0 and 1; $sim\ (x , y) = 1$ means that they are exactly the same in their properties ; $sim\ (x , y) = 0$ means that there are no common properties between the concepts. We compute the similarity here using the Dice distance fraction as follows:

$$Sim\ (x , y) = (2|x \cap y|) / (|x| + |y|) \quad (1)$$

where $(x \cap y)$ is the set of the common properties of the concepts, and $(|x| + |y|)$ is the sum of the properties size of the two concepts.

3.3. Intelligent Agent

IAs are systems that are situated in some environments and are capable of autonomous actions in this environment in order to meet their design objectives. Agents have some properties such as autonomy, intelligence, social-ability, reactivity and mobility [26]. There are two types of agents: mobile and static agents. Mobile agents can move within a network and act on behalf of the user or another entity. Mobile agents function independently or cooperatively to solve problems, while the static agent can function only locally. We define two new agents namely description and request agents.

Definition 5: Description Agent (DA) is a static agent that carries some information and automatically performs some set of functions and belongs to a grid node.

The carried information is needed for communication between the grid nodes. The DA functions are describing resource capabilities using the ontology, informing its neighboring DAs about its resources status as well as updating them when there is a change on the resource information.

Definition 6: Request Agent (RA) is a mobile agent that carries some information, automatically performs some set of functions and belongs to a grid node.

RA information consists of resource request and nodes information. RA generates resource requests for applications, and acts on behalf of the grid user by using ontology; RA then roams the network to find the node that owns the requested resource.

The reason behind using DA and RA is that DA is adaptive to the dynamic nature of the grid. In this context, when a node changes its resource status, neighboring nodes should be aware of that change, which in turn optimizes the request routing. RA helps the user to formulate his/her resource requests. This will ensure uniformity between resource descriptions and requests.

4. The Proposed RD Framework

The new RD framework initially consists of two aspects: description and locating of the resources. Description includes the ontology model that we have discussed in section 3.2. Meanwhile, locating includes the resource node organization and resource search process. In this section, we elaborate the locating aspect as well as its interaction with description.

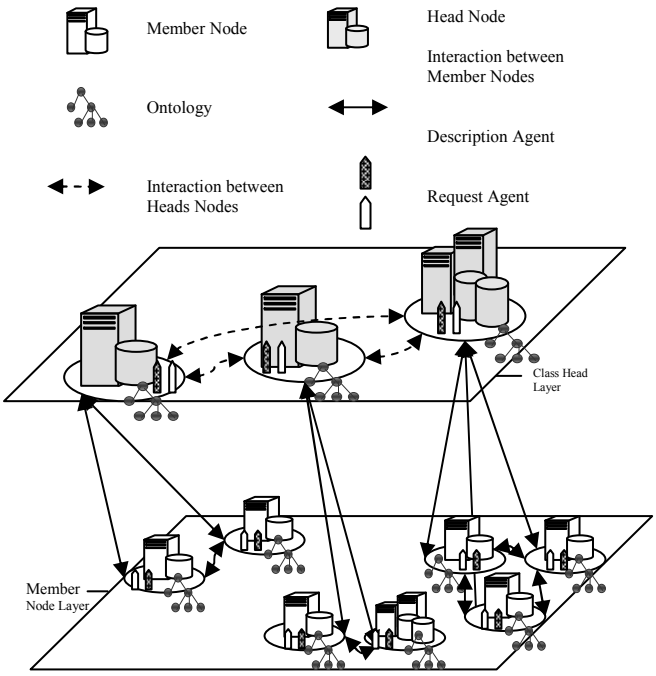

Figure 1. The conceptual model of the new RD framework.

4.1. The Node Organization

Node organization refers to the architecture of nodes that hold the information of resources (metadata). It composes of four components, which are class formulation, head appointment, node subscription and class maintenance.

4.1.1. Class Formulation

To support scalability and dynamism in grid environment, we model a grid system that contains some nodes to class based organization. This classification is based on some predefined criteria, such as resource interest among nodes, resource importance, and geographical location. For simplicity, we use the resource types as criteria for classifying the

nodes (e.g., computing resources, data resources, scientific instruments, *etc.*).

Definition 7: resource type is a collection of several grid resources that are identified based on their usability towards grid applications or services. For example, computing resource type can cover clusters, storage, and CPUs; data resource type may be databases, knowledge bases, online library service and so on.

With the aim of forming the classes, we create a class formulation algorithm that takes the predefined resource type features and matches them with the resource types of the given grid nodes. The algorithm finally returns a list of nodes for each class. The pseudo code of the algorithm is as follows:

Class Formulation Algorithm
Input : Resource type $t \in T = \{ t_1, t_2, t_i, t_d \}$, *node* $n \in N$, *Threshold;*
/ where N is set of the nodes & T is list of the resource types that is defined by a Grid community */.*
Output: listOfClassNodes c;
/ a list of class nodes that related to a resource type */*
Step 1:
FOR ($\forall n \in N$) *DO*
SimBetween (n , t)=$(2|n \cap t|) / (|n|+|t|)$
IF (SimBetween (n , t) > Threshold)
THEN DO
ADD (n , c)
/ calculate the similarity between the node and resource types, then, add into class list the nodes that have similarity degree higher than the defined threshold */*
END;
END;
Step 2:
Return c;

4.1.2. Head Appointment

The class formulation algorithm gets a set of classes corresponding to the defined resources types. Each class needs to have a head that will ease the communication between the different classes. This process is called head appointment, which consist of two steps. First, we need to define the headship features for which a node can qualify to be become a head. We suggest performance capabilities and node availability as the headship features. Performance capabilities are the speed of the server, network bandwidth, reserved memory space for resource information, and so on. Availability is the proportion of the time when the node is persistent in a grid system. In the second step, a head appointment algorithm calculates the similarity between the nodes and the defined headship features and selects the class head based on similarity degrees.

Head Appointment Algorithm
Input: ListOf class c, HeadshipFeatures HFea;
Output: ClassHead Head, SortedClassMembers mSort;
Step 1:
For ($\forall m \in c$) *DO*
GET Sim(m , HFea)
/ compute the similarity degree between the class node member m and the predefined headship features*/*
END;
Step 2:
SortedList = Sort (c, Sim(m,HFea))
/ sort class nodes according to their similarity degree*/*
HeadOfClass = $m \rightarrow max\ Sim(m,HFea) \in SortedList$
/ select the highest similarity degree node to be as head of the class*/*
Step 3:
Return HeadOfClass, SortedList;

The selected head maintains two kinds of information: a summary of the resource information of its class and resource type information of the other classes. The first type of information allows it to forward the resource requests to relevant node within the class. The second information helps in the forwarding of the resource request to the relevant class when the request is not related to the requester class.

4.1.3. Node Subscription

The first two processes of the node organization will create some classes with their heads. This section describes how a new node can subscribe to the grid system. Subscription is the procedure of assigning a new node to an existing class or set of classes that corresponds to its resource type. We design a node subscription algorithm for the subscription. The algorithm assumes that the new node is given the information about the grid resource type during the settings. The new node sends a message that contains its resource type to any existing nodes (members/ heads). The algorithm takes the resource type of the new node and calculates the similarity degree between the type and the related class heads; if the similarity degree attains the predefined threshold, the new node is added to the class of that head. Finally, the algorithm returns the list of the heads to which the node is assigned.

Node Subscription Algorithm
Input: NewNode nNew, ResourceType $T \rightarrow nNew$, *ExistingNode* $n \in c$, *Threshold;*
Output: ListOfHeads List;
/ list of heads that the new node is assigned to */*
1: Send message from nNew to $n \in N$
2: IF ($n \rightarrow HeadOfClass$) *THEN DO*
3: For ($\forall HeadOfClass \rightarrow \exists t_{nNew}$) *DO*
4: GET Sim (t_{nNew}, *HeadOfClass)*
/ calculate the similarity degree between the new node type with all Heads that associated with these types */*
IF (Sim (t_{nNew}, *HeadOfClass)> Threshold)*
THEN DO
ADD (nNew , HeadOfClassList)
END;
END;
5: ELSE forward the message to the HeadOfClass

Go to Step 3 & 4
Return List;

4.1.4.Class Maintenance

Grid node dynamism has an effect on the node organization. A class maintenance scenario to cope with this situation is essential. Class maintenance will take place in two cases: a failure of a class head and a failure of a class member. Both of these cases can take place in the grid system voluntarily or due to other connection problems. We propose two mechanisms to handle head replacement and member replacement.

4.1.4.1 Head Replacement

Existing heads are supposed to replicate their resource information to their predecessors in the headship ranking. Remember, a head is selected based on the similarity degree with the predefined headship features. Since the existing head has the highest similarity degree, a predecessor can be the second highest and so on. When the head wants to leave or fails the predecessor can replace it. The predecessor then (new head) informs its class nodes about itself and performs all the functions of the previous head.

Head Replacement Algorithm
Input: MessageTime t, ClassHead Head, PredecessorHead PHead;
while (time = t) DO
Head send nodeInfo→PHead;
If ((time>t ∧ !nodeInfo) ∨ leaveMessage) THEN DO
For (∀ Heads ∧ calassNode)
Inform about the new head;
send nodeInfo→PHead;
END;

4.1.4.2 Member Replacement

Member replacement can be achieved by connecting the direct neighbors of the withdrawn member. As each class is a connected graph, each member has connection to two neighbors and its leader. Member is supposed to inform its back neighbor about the front one, and likewise the front about the back. When the member in between the back and front members is dropped, the two remaining members will fill the gap through their connections.

Member Replacement Algorithm
Input: MessageTime t, ClassNode Node, RightNeighbor RNode, LeftNeighbor LNode
while (time = t) DO
Node sends nodeInfo → RNode ∧LNode;
If ((time>t ∧ !nodeInfo) ∨leaveMessage) THEN DO
Node sends RNodeInfo to LNode;
Node sends LNodeInfo to RNode;
END;

4.2. The Resource Discovery Process

Figure 1 shows our overall conceptual model of the framework, which discussed the previous sections. In this section we discuss the resource discovery process, and on how a resource request can be formulated and processed. The nodes are organized in graph form that is formalized as: $G = (V , E)$, $V = \{ v_1 , v_2 , v_i , \ldots v_n \}$, $E = \{ e_1 , e_2 , e_i , \ldots e_x \}$, where, V represents the set of the nodes and E the set of connections between nodes. Since a grid has a set of resources and applications, we form two vectors. The first one is for the resources, and the second is for the applications $R = [r_1 , r_2 , r_i , \ldots r_n]$ and $A = [a_1 , a_2 , a_i , \ldots a_n]$, where, r_i is a shared resource, and R is the overall resources on a grid; a_i is a grid application that requires a resource r_i. Note that, resources and applications are described semantically using ontologies that we have mentioned in section 3.2. Based on the two vectors (resource and application), we can construct an adjacency matrix called *job matrix J(A×R)*.

$$\therefore \quad J (A \times R) = \begin{pmatrix} a_1 r_1 & \cdots & a_1 r_k \\ \cdots & a_i r_j & \cdots \\ a_n r_1 & \cdots & a_n r_k \end{pmatrix}$$

where the J element $a_i r_j \in [0,1]$, $a_i r_j = 1$ when an application a_i requires r_j resource and 0 otherwise. Another adjacency matrix formed is called the resource node matrix M, which is based on the resource and grid node vectors $M(N \times R)$.

$$\therefore \quad M = M (N \times R) = \begin{pmatrix} n_1 r_1 & \cdots & n_1 r_k \\ \cdots & n_i r_j & \cdots \\ n_n r_1 & \cdots & n_n r_k \end{pmatrix}$$

where the M element $n_i r_j \in [0,1]$, $n_i r_j = 1$ when node n_i has the resource r_j and 0 otherwise. It should be noted that, an adjacency matrix from nodes of a class and a set of resources is a sub matrix of the resource node matrix M. Each node may have a Job matrix and sub resource node matrix (m). The job matrix helps the user to create his/her resource request, while the resource node matrix maintains information about the node of the resources.

To allocate resources for user's tasks we develop an algorithm that searches resources on the network based on local information and dynamic matching. Local information is the presence of particular resources in a node, which is described in the sub resource node matrix. Dynamic matching is the similarity calculation between agents that represent resource provider and requester using the similarity function of section 3.2.

Resource Search Algorithm
Input: NodeInformation , ResourceRequest, Threshold.
Output: ListOfNodes List;
Step 1:
Get NodesInformation Form DA
Step 2:
IF (NodeInformation → neighboringNode) THEN DO
For (∀neighboringNode → requstedResources)
Get Sim(RA, DA)

IF Sim(RA, DA) $\geq$ *Thrashold Then Do*
ADD(neighboringNode, lisT)
ELSE Send ResourceRequest to ClassHeads
FOR($\forall$ *ClassHeads* $\in$ *Class* $\rightarrow$ *requstedResourcesType)*
IF ($\exists$ *ClassNode* $\rightarrow$ *requstedResources) Then DO*
FOR ($\forall$*ClassNode* $\rightarrow$ *requstedResources)*
Get Sim(RA, DA)
IF Sim(RA, DA) $\geq$ *Thrashold Then Do*
ADD(ClassNode, lisT);
END;
END;
Step3:
Return List;

5. Application

In order to clarify the interactions among our system components in describing and discovering a resource using the resource search algorithm, we introduce an example that shows how a grid user can describe or request a resource.

Assuming that we build a grid with 1024 nodes distributed in different locations. Using the ontologies model of section 3.2, we can define the resource types, say 4 types, each type may contain 32 resources, thus, the total number of the resources is 32*4= 128. Initialize the two agents (DA and RA) in each node. Implementing the class formulation and head appointment algorithms respectively on the nodes or DAs (nodes are represented by their DAs), we obtain a set of classes with their heads. For simplicity, we may have 4 classes since the resource types are 4, and each class has 256 nodes. Each DA sends its resource information to its two neighboring DAs as well as the head. Accordingly, the head will have the entire information summary of class, which will be a sub resource node Matrix (*m*). If new nodes want to subscribe to the system, the node subscription algorithm in section 4.1.3 is activated. Moreover, if an existing class member or head node quits the system, the mechanism of class maintenance in section 4.1.4 will manage the exit. Assuming a user wants to run some applications. The steps to request and discover the resources according to our new framework are as follows:

a. Based on the local ontology, the user selects an application a_i from the set of applications $A = [a_1, a_2, a_i, \ldots a_n]$; the job matrix enables RA to form a resource request vector using J, say 16 resources.
b. From the local information given by the DAs (sub resource node matrix), RA sends a request to any neighboring node n_i that is associated with all or part of the 16 resources requested and the threshold of the similarity degree.
c. Based on the semantic description of resources in RA and DA, the similarity degree of the two agents *sim(RA,DA)* is calculated with regards to the resource properties of the requested resource and provided resource.
d. If the similarity degree of *sim(RA,DA)* reaches a user defined threshold value, then select the node n_i and check whether there are still remaining requested resources to be searched.
e. Repeat steps c and d until there are no nodes in the class associated with the requested resources.
f. If so, then send the remaining requested resources to a class head c_i.
g. From the resource node matrix M the head c_i sends the request to another class head/heads c_j that may have the remaining requested resources.
h. For each head, repeat steps b, c, d and e until all the 16 requested resources are found.

6. Discussion

As we have mentioned in the introduction that any new grid RD mechanism is expected to have some features such as: scalability, decentralization, dynamism and interoperability. This section discusses how the new framework meets these characteristics.

Scalability in grid is the increase of the number of resources or users that use the resources. Our framework is scalable by using the class based node organization, which gives an opportunity to any class to grow upwards rather than treating all the nodes as one group, as is the case of the current centralized RD systems. To further proof the scalability, we elaborate the above application by using different number of resource types and calculate the number of the query hops that are needed to discover a resource in the system. Table 1 describes the quantity of the nodes, resource types, class sizes and the requested resources. We focus on the number of nodes and requested resources, and vary the resource type from 4 to 64. In each resource type, we calculate the average and maximum number of the query hops. Query hops are the number of message forwarded from the *requester* (the node from which a resource request is sent) to other nodes until the request is satisfied.

Table 1. The quantity of the application components.

Nodes	Number of Resource Types	Number of Nodes for each Class	Number of the Requested Resources
1024	4	256	16
1024	8	128	16
1024	16	64	16
1024	32	32	16
1024	64	16	16

Class size is the number of the nodes within a given class. To get the size of each class, we divide the total number of nodes by resource types. For example, in the second row of Table 1, the resource types are 4 so we get 256 for each class.

Table 2. The notations of the application components.

Symbol	Description
N	number of nodes
T	number of resource types/ class
R	set requested resources has one or more r elements
t	Number of the requested resources in each resource type
δ	number of nodes for each class/ class size
λ	number of the requested resources
β	number of the requested resources in each class
ρ	maximum query hops to find a resource r
μ	overall maximum query hops for T
θ	overall average query hops for T

To simplify the presentation, we denote the components of Table 1 and other variables that are needed to be calculated in Table 2. Based on the above notation, we form the query hops equations as follows:

$$\rho = r * \delta \tag{2}$$

$$\mu = \sum_{r=1}^{R} \rho \tag{3}$$

$$\theta = \sum_{r=0}^{R} \frac{\rho}{2} \tag{4}$$

Referring to definition 7, a resource type may include a set of resources. This means, the requested resources may belong to one or many different resource types. Therefore, we can have this fact. $\because R = \{ r_1, r_2 , r_i , \ldots r_n \}$ and $T = \{\{t_1\}, \{t_2\} , \{t_i\} , \ldots \{t_n\}\}$ $\therefore R \subseteq T$, Using the above fact, the 16 requested resources could be equally assigned to the different resource types. The resource type of the requested resource is calculated as follows:

$$\tau = \begin{cases} \frac{R}{T}, & R \geq T \\ R, & R < T \end{cases} \tag{5}$$

We implement the above equations on the given data of Table 1. For each number of the resource types (number of the classes), we get the maximum and the average number of the query hops. Figure 2 verifies the relation between the number of classes and query hops.

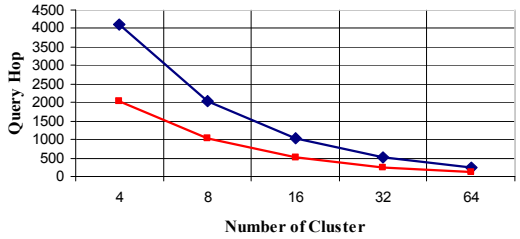

Figure 2. The query hops of the requested resources.

The x axis represents the number of classes while the y axis represents the query number of query hops. It is observed on the curves that μ (blue) and θ (red) start with high values when T is small and decrease gradually until they reach to their minimum value when the number of classes is 64. Based on this graph, two conclusions can be made. First, the number of the classes is crucial in limiting the scope of the query hops. For example, when the number of the classes is 8, μ is 2048 hops. Meanwhile, when the number of the classes is 64, μ is 256 only. This shows that whenever the class number increases, the scope of resource look up gets smaller. Secondly, the new framework can be more scalable if there is a mechanism for identifying the number of the classes so that the proportion of the query scope can be smaller.

Decentralization in grid RD is keeping resources information not under a common server control as in Condor [4] system. In our case, each node maintains its own resource information and each head node maintains a summary of other classes' information (resource types). This means that no node replies a query on behalf of the other. In addition to that, we show how a head node can be succeeded when it wants to leave or fails. In this way, the system may not be completely down as in centralized systems. In the current RD systems, a node can answer on behalf of the others.

Dynamism is about tracking the status of the resources as they are dynamic and can come in and move out from the grid system. We use intelligent agents, DAs to track the status of each node resources, which in turn allows the nodes to update their sub resource nodes matrices m.

Interoperability is the ability of a RD system to span multiple administrative domains in discovering the resources. The use of ontology in this framework allows a well-defined meaning to resource information and provides a uniform description and discovery semantic among the participants. There will be no syntactic matching as in the case of the current systems.

In addition to the above features, the framework provides fault tolerance. In this case, the system can tolerate the failure of member nodes and class heads which has been described in section 4.1.4.

7. Conclusions

In this paper, a novel framework for grid resource discovery is presented. The framework consists of two aspects: description of the resource information and the look up of the resource information. The paper proposes an ontology model to describe resources, applications and their relationships. P2P network architecture and intelligent agents are integrated for the resource look up process. In this context, the JXTA architecture is mapped to organize the nodes. Nodes are organized in some classes. Each class has a head that eases the communication with other heads. Several algorithms are presented for creating classes,

appointing heads and maintaining the grid system. Intelligent agents are used for advertising resource capabilities and creating resource requests. We present an application that shows how the framework handles the resources discovery process. Through the discussion and calculations, we have shown that the framework satisfies the RD characteristics, which are scalability, decentralizations, dynamism, and interoperability.

References

[1] Andrzejak A. and Xu Z., "Scalable, Efficient Range Queries for Grid Information Services," *in Proceedings of the Second International Conference on Peer-to-Peer Computing*, pp. 33-40, 2002.

[2] Condor Project: http://www.cs.wisc.edu/condor/, Last Visited 2009.

[3] Cai M., Frank M., Chen J., and Szekely P., "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," *in Proceedings of 4th International Workshop on Grid Computing*, pp. 184-191, 2003.

[4] Chandrasekaran B., Josephson J., and Benjamins R., "What are Ontologies, and Why Do We Need Them?," *Computer Journal of IEEE Intelligent Systems*, vol. 14, no. 5, pp. 20-26, 1999.

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

[6] Flahive A., Taniar D., Rahayu W., and Bernady O., "Ontology Tailoring in the Semantic Grid," *Computer Journal of Standards and Interfaces*, vol. 6, no. 3, pp. 282-284, 2008.

[7] Globus Toolkit, http://www.globus.org.

[8] Hassan M. and Abdulah A., "Scalable Self-Organizing Model for Grid Resource Discovery," *in Proceedings of International Conference on Network Applications, Protocols and Services*, Malaysia, pp. 426-429, 2008.

[9] Han L. and Berry D., "Semantic-Supported and Agent Based Decentralized Grid Resource Discovery," *Computer Journal of Future Generation Computer Systems*, vol. 24, no. 8, pp. 806-812, 2008.

[10] Hassan M. and Abdulah A., "Semantic Based Scalable Decentralized Grid Resource Discovery," *in Procedeings of International Conference on e-Technology Singapore*, pp. 3316-3324, 2009.

[11] Iamnitchi A. and Foster I., "On Fully Decentralized Resource Discovery in Grid Environments," *in Proceedings of the Second International Workshop on Grid Computing*, Colorado, pp. 51-62, 2001.

[12] Lacasta J., Nogueras-Iso J., Be´jar R., Muro-Medrano P., Zarazaga-Soria F., "A Web Ontology Service to Facilitate Interoperability Within a Spatial Data Infrastructure: Applicability to Discovery," *Computer Journal of Data and Knowledge Engineering,* vol. 63, no. 5, pp. 947-971, 2007.

[13] Meshkova E., Riihijärvi J., Petrova M., and Mähönen P., "A Survey on Resource Discovery Mechanisms, Peer-to-Peer and Service Discovery Frameworks," *Computer Journal of Networks*, vol. 52, no. 2, pp. 2097-2128, 2008.

[14] OWL, http://www.w3.org/2004/OWL, Last Visited 2009.

[15] Oaks S., Traversat B., and Gong L., *JXTA in a Nutshell*, O'Reilly, 2003.

[16] Padmanabhan A., "SOG: A Self-Organized Grouping Infrastructure for Grid Resource Discovery," *PhD Thesis*, University of Iowa, Iowa, 2006.

[17] Padmanabhan A., Wang S., Ghosh S., and Briggs R., "A Self-Organized Grouping Method for Efficient Grid Resource Discovery," *in Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pp. 312-317, Washington, USA, 2005.

[18] Puppin D., Moncelli S., Baraglia R., Tonelotto N., and Silvestri F., "A Grid Information Service Based on Peer-to-Peer," *in Proceedings of the 11th International Euro Par Conference*, Portugal, pp. 454-464, 2005.

[19] Parkin M., Burghe S., Corcho O., Snelling D., and Brooke J., "The Knowledge of the Grid: A Grid Ontology," *in Proceedings of the 6th Cracow Grid Workshop*, Poland, pp. 658-662, 2006.

[20] Perez A., Sanchez A., and Abawajy J., "An Agent Architecture for Managing Data Resources in a Grid Environment," *Computer Journal of Future Generation Computer Systems*, vol. 1, no. 1, pp. 747-755, 2008.

[21] Ranjan R., Harwood A., and Buyya R., "Peer-to-Peer Based Resource Discovery in Global Grids a Tutorial," *Computer Journal of IEEE Communications Surveys and Tutorials*, vol. 10, no. 6, pp. 6-33, 2008.

[22] Shen H., "A P2P-Based Intelligent Resource Discovery Mechanism in Internet-Based Distributed Systems," *Computer Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 197-209, 2009.

[23] Schoder D., Fischbach K., and Schmitt C., *Peer-to-Peer Computing: The Evolution of a Disruptive Technology*, Hershey Idea Group Publishing, 2005.

[24] Sun Microsystems, http://www.sun.com/, Last Visited 2009.

[25] Trunfioa P., Taliaa D., Papadakisb H., Fragopouloub P., Mordacchinic M., Pennanend M., Popove K., Vlassovf V., and Haridi S.,

"Peer-to-Peer Resource Discovery in Grids: Models and Systems," *Computer Journal of Future Generation Computer Systems*, vol. 23, no. 6, pp. 864-878, 2007.

[26] Wooldridge M., *An Introduction to MultiAgent Systems*, John Wiley and Sons, 2006.

[27] Xing W., Dikaiakos D., and Sakellariou R., "A Core Grid Ontology for the Semantic Grid," *in Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid,* Singapore, pp. 178-184, 2006.

[28] Zhang Y., Qu Y., Huang H., Yang D., and Zhang H., "An Ontology and Peer-to-Peer Based Data and Service Unified Discovery System," *Computer Journal of Expert Systems with Applications*, vol. 36, no. 3, pp. 5436-5444, 2009.

Mahamat Hassan is a PhD candidate in computer science and graduate assistant at the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia. He received his BEng in computer engineering in 2003; and MSc degree in computer science in 2005. His research interests include sharing resource over internet systems such as grid, peer-to-peer, web services and so on, and the use of ICT in developing countries.

Azween Abdullah is an associate professor in the Department of Computer and Information Sciences at Universiti Teknologi PETRONAS, Malaysia. He obtained his BSc in computer science in 1985, MSc in software engineering in 1999, and PhD in computer science in 2003. His work experience includes twenty-one years in institutions of higher learning and commercial companies.