International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(4): 848-865
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Semantic Technology and Super-peer Architecture for Internet Based Distributed System Resource Discovery

Mahamat Issa Haasn
Department of Computer and Information Sciences
Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia
mahamat.hassan@petronas.com / mhtissa@gmail.com

ABSTRACT

Resource/service discovery is a very vital issue in Internet based distributed systems such as Grid and Cloud computing. In this paper we address the RD issue in intergrid. We design a service discovery framework by integrating semantic technology, peer-to-peer network and intelligent agents. The framework has two main components which are service description, and service registration and discovery models. The earlier consists of a set of ontologies that are used as a data model for service description and services to accomplish the description process. The service registration is based super-peer architecture to organize the nodes and on ontology to manage the node organization. In addition to that, we introduce intelligent agents to automate the discovery process. We evaluate the framework via simulation experiments, and the result confirms the effectiveness of the framework in satisfying the required RD features (interoperability, scalability, decentralization and dynamism).

KEYWORDS

Grid; Cloud; semantic technology; intelligent agent; and peer-to-peer network.

1 INTRODUCTION

Internet based distributed systems such as Grid [1] and Cloud[2] computing have become a vital resource sharing infrastructure for today's scientific and business applications. One of the fundamental components of theses infrastructure is *Resource/Service Discovery* (RD) which is about the detection of suitable resource for a given task/application. This paper addresses the service discovery issue in the case of intergrid. Intergrid/Global grids (in some literatures is also called multi-grid [3]) are known as a grid of grids, since they are a collection of small grids that cross organizational boundaries to create very large virtual systems that can be accessed from anywhere in the world. Generally, RD process entails *description of the resource through its properties*, *registration/indexing of the described resource in common registry(s)*, and *discovering the registered resources that match with resource request specifications*. These steps correspond to the main components of the RD system, which are *Description*, *Registration* and *Discovery (*which is composed of *search* and *selection)*. Unfortunately, intergrid systems are normally associated with some complexities such as resources/services and users are

distributed across different locations; resources are heterogeneous in their platforms; status of the resources is dynamic (resources can join or leave the system without any prior notice); and use of multi-middleware. These complexities pose a challenge to the development of an efficient RD system to discover the resources and services. In fact, these complexities also yield some requirements that should be fulfilled by any developed RD. These requirements include *high searchability* (interoperability) to retrieve the relevant and precise resources and services, and *high performance (scalability, decentralization, and dynamism)* to make the RD system sustainable with the scale of the intergrid. In fact, there is a wealth of work on grid RD (e.g. Globus[1], Condor[2] , [4], [5], and [6]) that we have currently but they have difficulties to attain these features. Therefore, In this paper, we introduce a new intergrid RD framework that can overcome the shortcoming of the current studies and meeting the above mentioned requirements. The framework contains two main components which are *service description,* and *service registration and discovery* models. The earlier consists of a set of ontologies and services. Ontologies are used as a data model for service description, whereas the services are to accomplish the description process, we detail that in section 2.

The service registration is also based on ontology, where nodes of the services (service providers) are classified to some classes according to the ontology concepts, which means each class represents a concept in the ontology. Each class has an elected head. Head plays the role of a registry in its class and communicates with the other heads of the classes in a peer to peer manner during the discovery process. We further introduce two intelligent agents to automate the discovery process which are *Request Agent* (RA) and *Description Agent* (DA). Each node is supposed to have both agents. DA describes the service capabilities based on the ontology, and RA carries the service requests based on the ontology as well. We design a service search algorithm for the RA that starts the service look up from the class of request origin first, then to the other classes, we detail that in section 3.

We finally evaluate the performance the framework with extensive simulation experiments, the result of which confirms the framework effectiveness in satisfying the required RD features (interoperability, scalability, decentralization and dynamism), we detail that in section 4.

In short, our main contributions are an interoperable semantic description RD component model for intergrid services metadata representation; a semantic distributed registry architecture for indexing service metadata; and an agent-based service search and selection algorithm.

2 SEMANTIC-BASED RESOURCE DESCRIPTION MODEL

2.1 The Model Description

In order to have a description model that meets RD requirements, we refine the intergrid system in such a way that makes full use of the resources and services when the semantic technology is applied. A common ontology is used

[1] http://www.globus.org/.

[2] http://www.cs.wisc.edu/condor/.

to formally represent the intergrid components.

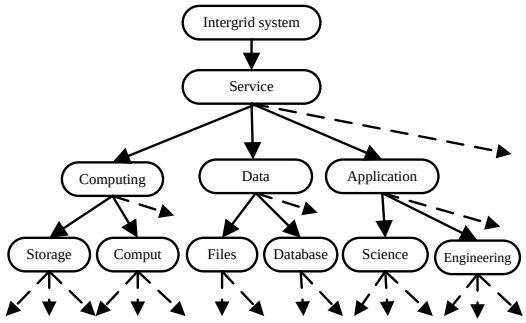


Fig. 1. Fragment of service grid domain ontology

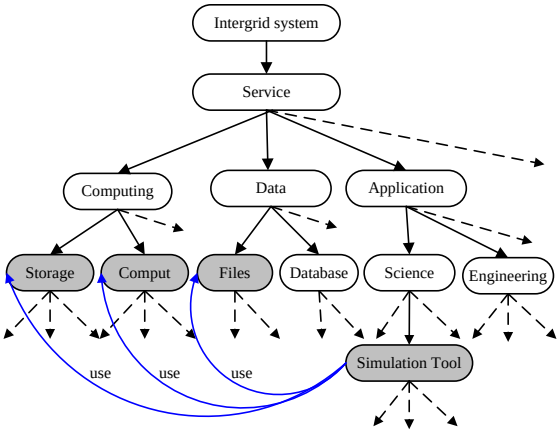


Fig. 2. The extraction of application goals from the service grid domain ontology

Subsequently, we merely rely on the latest grid system requirements that have been presented by the OGF[3] [7] in defining the grid level and intergrid level. As a result, we treat grid level system as a *service grid* that is provided by a *provider* to *consumers*, and this service grid is assumed to be among the grid types (e.g. computing, data, and application). Therefore, an intergrid level system is a collection of service grids that have agreed to work cooperatively as consumers and providers. Consequently, a service grid may be a consumer as well as a provider. It becomes a consumer when it uses other service grids without providing any service to them, whereas it functions as a consumer/provider when other services are added to its own and at the same time it also provides a complete service to the end user. The capabilities of service grids are described by aggregating their local metadata content and then integrating them into a common information model. Ontology ([8] and [9]) is used for the common information model. We call this ontology as *service grid domain ontology (SGDO).* SGDO defines all the service grid types, attributes that are needed for each service grid, relationships between all the services, structure of the values of each attributes and so on (see Fig. 1). To reduce the user interaction with programming details with RD system in specifying the service grid requests, we introduce a mechanism that is called Goal-based Service Grid Request Description (GSGR). A goal refers to what a given consumer/end user wants to achieve by using the service grids. For example, if a user wants to simulate the weather condition of the earth so the simulation is his/her goal. Obviously, a goal requires a set of the grid services in order to be accomplished. For example, the simulation of weather condition of the earth requires computing service grid, satellite images data and temperature dataset which can be under the data service grid and so on. The SGDO, among other concepts of application service grid, includes all software applications that are available on the intergrid level system. In fact, these applications represent the goals that a user may want to achieve because application service grids are the only services that need one or more service grids to work on, as they cannot stand

[3] Open Grid Forum: http://www.ogf.org/

alone. Thus, we can extract the goals from the SGDO. We introduce a relation so called "*use"* between the application service concepts and the other service grid concepts. The "use" relation is a binary relation between a particular application service concept and another service grid (e.g. data service). Indicating this application service requires the second service grid with which the relation is established (see Fig. 2).

2.2 The Model Building Block

Having described the fundamental components of the description model, in this section we illustrate the model building block. Fig. 3 shows the components of the model and their related subcomponents. The model is initially composed of Semantic Description Manager (SDM) and Service Grid Metadata Provider (SGMP). SDM generally is responsible of the global service grid description in the intergrid system, and a pool that accommodate the service grid metadata coming from SGMPs. Meanwhile, SGMP is responsible of managing local service grid metadata that belongs to a service grid provider. The reason for having SDM and SGMP in such architecture is that, SDM will provide all the needed information and data model management for a set of intergrid members. Therefore, interoperability can be ensured. In the meantime, SGMP provides autonomy to each service grid member as it handles the local information of the service grid.

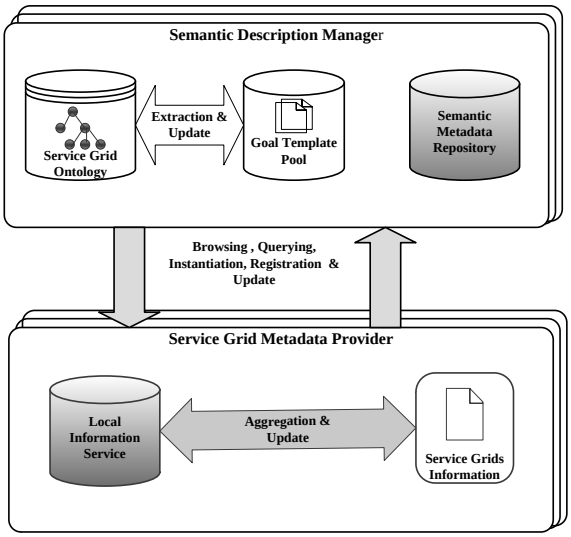


Fig. 3. The description of model building block

2.3 The Description Process

The description process includes description of a service that will be advertised, and service request formulation. The steps of the first case are as follows:

- The user invokes the SGMP system.
- The user browses/queries the service grid ontology through which all the available content of the ontology can be manipulated (e.g. using add and drag menu), and selects the concept that is relevant to his/her service grids.
- The user gets an instance of the selected service grid concept using the service grid ontology tools.
- The user populates the instance with the actual service grid information, which is an aggregated summary of the overall service grid information.
- Finally, the user sends the service grids information to the respective SMR of the SDM node that is responsible of holding the metadata of the current service provider, and in turn, the SMR stores the semantic

information about the service grid for the discovery process.

Meanwhile for the service request formulation is as follows:

- The user invokes the SDM system.
- The user browses/queries the available goals in the goal template manager, and selects the relevant goal.
- The user then gets an instance of the selected goal and adds it to his/her local information system.
- Since each goal requires one or a set of service grids, the user adds the concrete values of attributes of each service. For example, if among the required service is computing service, and one of the attributes is maximum number of computing nodes, the user may add a concrete number for such requirements.
- Finally, the user sends the service grid request to the respective SDM, and the SMR of that SDM will generate a proper query statement for each service among the required service grids.

2.4 Model Evaluation

From the building block, it is clear that the model has introduced the use of semantic information in way that does not require the use of local information service, which exists currently in grid middleware. For example, the intergrid participants (small grids) are able to use their local discovery system that would normally be possible through a keyword-based RD system. They just need to have one file that accommodates the summary of the overall capabilities of the service. As a result, this provides interoperability among the participants in the intergrid system. The model also reduces the cost of using semantic information in terms of processing time, as well as storage of the semantic information, since the semantic information is used at the intergrid level, and not at the grid level. Therefore, during the discovery we look up for a complete service grid, not components of it. Therefore, the model achieved the first RD requirement (high searchability).

3 SEMANTIC REGISTRATION AND DISCOVERY MODEL

Registration and discovery components in any RD system are very much related, as the routing of request is subjected to the registration architecture. For this reason, we address the issues in registration and discovery jointly. We design a model for the two components that integrates super-peer architecture, ontology and intelligent agent. Super-peer is used to grant distribution of the registry where the service grid metadata is located.

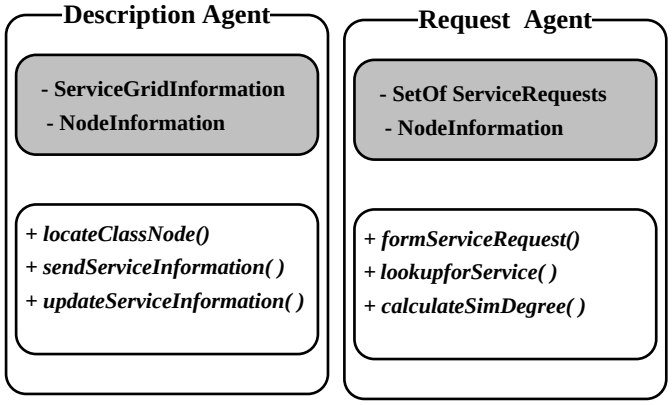


Fig. 4. The proposed DA and RA agents

3.1 The Model Components

The model consists of three components, domain ontology, an intelligent agent model, and super-peer architecture. Ontology concepts are normally arranged hierarchically, therefore, whenever, we visit the concepts from the root concept, as we go down deeper into the subconcepts, we will move from a more general class of concepts to a more specific class of concepts, and vice versa. We use this feature to classify nodes into several classes, which produce registry architecture to the RD system. The ontology that supplies service grid taxonomies is called Dictionary Ontology (DO). The DO may be the same as the service SGDO by omitting the relations that are out of the hierarchical relation such as the "use" relationship. Fig. 4 shows the proposed agent where *DA* is a static agent that carries some information; automatically performs some set of functions and belongs to a service grid provider node. RA is a mobile agent that carries some information; automatically performs some set of functions and belongs to a service grid node.

3.2 The Model Description

The registration and discovery model consists of three elements registry architecture, fault tolerance and load balancing strategy, and discovery algorithm. In this section we discuss all these elements.

a) Registry architecture

The registry architecture includes node class formulation, head appointment, node subscription. In class formulation, nodes are gathered together in a set of classes. This classification is based on the hierarchal relations among the service grids in the DO, which means their defined semantic relation on the DO. For example, nodes that provide service grids that belong to the computing concept in the DO can form a class of nodes called computing class. We design an algorithm to accomplish the class formulation. In head appointment, each class needs to have a head that will ease the communication between the different classes. In this process, we first need to define the headship features, for which a node needs to qualify to become a head. In the second step, a head appointment algorithm calculates the similarity

between the nodes and the predefined headship features, and selects the class head based on the degrees of similarity. An algorithm to perform the head appointment is designed. Node subscription refers the procedure of assigning a new node to an existing class or set of classes that corresponds to its service concept. Subscription is done by the node subscription algorithm. In this algorithm, we assume that the new node has been given the information about the selected service grid concepts during the settings, and the new node sends a message that contains its service concept to any existing node (member/head). The algorithm takes the service concept of the new node, and calculates the similarity degree between the service concept 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 heads, for which the node has been assigned to, if the new node has more than one service grids that belong to different concepts. More details about the three mentioned algorithms can be found in [10].

b) The fault tolerance and load balancing strategy

The fault tolerance and load balancing strategy address the issues of dynamicity of the service grid nodes status, and the management of the node of classes in terms of the number of classes and size of each class. We incorporated two approaches respectively. The first one is called class maintenance which deals with a situation of failure in a class head and failure of a class member. The approach replaces the respective failed node (head/member) with another node (detail about that can be found in [10]).

In load balancing strategy, the classes of the service provider nodes are supposed to be managed by their respective heads (e.g. hosting the service grid metadata of the class members). This management process involves a huge amount of messages due to the intraclass and interclass communications. As a matter of fact, if we do not have optimization strategy to manage this tremendous amount of traffic, we will eventually be in a situation of bottle neck in the head. We use the idea of having few hundreds of nodes to be managed by one head. Therefore, we can define a variable called max number of nodes (μ) in a class to control the number of nodes in the class. The number of classes in a given intergrid starts by selecting the most general concepts in the DO, then when the nodes under a particular class (concept) has reached μ, we split the concept by selecting a number of more specific subconcept. This will ensure that every class can grow smoothly with a balanced management in the heads.

c) The discovery algorithm

By assembling the above framework components, we will have an intergrid that has a set of nodes assigned to some classes with their heads. The collection of heads forms a head node layer, whereas the collection of classes and members forms the member node layer. Each node has two agents (DA and RA), and implements the SGMP element to describe their service grid information. Communication between the nodes will be through the exchange of messages between the agents. In addition to service SGMP, heads implement the SDM element to assist members to describe and register their services. Neighboring nodes in each class exchange information about their

services so that each node will have local information. Each head will hold the entire information of its class service as they are sent by the member nodes. In addition to that head is supposed to have some information about the other heads which includes their classes' concept.

The discovery algorithm addresses the search of the intergrid services on the network based on the cached information and dynamic matching. The cached information is the presence of a particular service in a node, which is got through the information exchange. Dynamic matchmaking is the similarity calculation between agents that represent service provider and requester, using the similarity function. The algorithm works as follows:

a) Based on the goals that are stored in the service goal template of the semantic description manager or the head of that user' node, the user selects the preferred goal and obtains instance of the goal. The user then adds the actual values of the service capabilities, which enable the RA to form a service request vector(s), say 6 services.

```
Input: NodeInformation, ServiceRequest, Threshold;
Output: ListOfNodes List;
Step 1:
Get GoalInstance
Add ConcreteValue to ServiceRequest
Get NodesInformation Form DA
Step 2:
IF (NodeInformation   neighboringNode) THEN DO
     For ( neighboringNode   requstedServices)
          Get Sim(RA, DA)
          IF (Sim(RA, DA)  Thrashold) THEN DO
          ADD(neighboringNode, list)
ELSE Send ServiceRequest to ClassHeads
   FOR(   ClassHeads   Class
          requstedServiceConcept)
  IF ( ClassNode   requstedServices)
                THEN DO
   /*send the service request to all the heads that are
   available in the current head local info and
   associated the service request concepts*/
       FOR ( ClassNode   requstedServices)
          Get Sim(RA, DA)
          IF Sim(RA, DA)  Thrashold THEN DO
          ADD(ClassNode, lisT);
Step3:
Return List;
END;
```

Fig. 5. The Discovery Algorithm

b) If there is local information about some neighbouring nodes that has been given by their DAs, RA sends a request to any neighbouring node n_i that is associated with all or part of the 6 requested services and the threshold of the similarity degree.

c) Based on the description of services in RA and DA, the similarity degree of the two agents sim(RA, DA) concerning the service properties of the requested service and provided service is calculated.

d) If the similarity degree of sim(RA, DA) reaches a user defined threshold value, then the node n_i is selected; and the check is done whether there are still remaining requested services to be searched.

e) If there are remaining service request, steps c and d are repeated until none of the nodes in the class is any longer associated with the requested service.

f) If so, then the remaining requested services are sent to a class head c_i.

g) From the head information, head c_i sends the service request to another class head/heads c_j that may have the remaining requested service based on the concepts.

h) For each head, the steps (b), (c), (d) and (e) are performed until all the 6 requested services are found.

Fig. 5 illustrates the discovery algorithm. It should be noted that, the above steps that represent the invocation of the new RD system.

4 RESULTS AND DISCUSSION

In this section we present a comprehensive quantitative evaluation with respect to the overall performance of the proposed RD framework. We have chosen one of the P2P simulators called PeerfactSim.KOM [11] to simulate the intergrid environment with the application of the proposed system. The evaluation of the system is based on some common performance metrics found in the literature [12] and [5]. This includes the percentage of the discovered services in a given goal request (Request/query hit), and the response time for the service request to be answered. These metrics are calculated in different settings of the nodes and service requests. Therefore, we start with a few numbers of nodes, and scale them gradually to simulate the increase of the services in the actual intergrid system. We also vary the rate of service requests from small number of requests to bigger number to simulate the increase of users in the intergrid system. We analyze the results of the different settings by highlighting the causes of the effects of the different setting to the results.

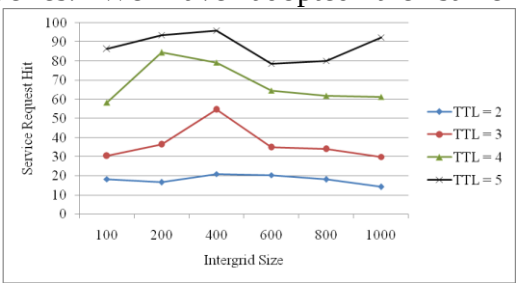
4.1 Experimental setup

We build an intergrid system that consists of n nodes. The size of the nodes n is scaled from 100 to 1000 with scale of 100 and 200. Since the creation of service grid domain ontology and dictionary ontology are out of our scope, we simulate these ontologies by representing them numerically. Where the concepts of the ontology are simulated by positive integer values such as 1, 2, …k, and each concept has subconcepts/properties which are some predefined set of values. Based on that, the concepts are representing the services' concepts and the predefined

values are representing the services themselves. Each of the nodes has a set of the services. The number of these services is varied between 1 and 6 services. The reason of having this range of numbers is that our RD system is based on aggregating the service grid resources and services metadata information, which obviously reduces the range of services in the intergrid system. The allocation of the size of services in each node is random, which is the same as the assignment of concepts to node. This is to simulate the fact that in intergrid system we may not be able to neither express precisely the number of the services in each node nor the type or the concept to which these services belong, but surely we can define the concepts in the first place. During the simulation of class formulation, each of these nodes will be joining a particular class based on the randomly assigned concept. The number of class is based on the number of concepts (if the selected concepts for the overall nodes are five, the corresponding number of the classes is also five). The selection of concepts is proportional to the size of the intergrid system. This is in correspondence to the super-peer architecture where the number of super peers is based on the size of the network. In [13] the number of super-peer node is implemented to be 5% of the nodes that have very high capacity to handle queries. We have adopted the same percentage so as to systemize the distribution of the node to classes in way that allows us to conveniently discuss the performance of the system. Therefore, an intergrid system that has a size of 1000 nodes will have 10 classes. For simplicity, during the simulation, nodes that have high capacity which are supposed to be the heads of classes will join the network first and declare themselves.

4.2 Performance of the new framework

We conducted 16 (this number corresponds to the variation of service request generation and TTL values) independent experiments for different service request portions and intergrid sizes. In each experiment, the mechanisms and algorithms that we have designed mentioned above are simulated. We first start our evaluation with the first performance metric, which is the service request hit. Fig. 6-9 show the simulation results for service requests generated by the nodes in percentages of 25%, 50%, 75% and 100% of the actual size of the intergrid. We control the forwarding of the request message from the requester node to the provider by the TTL values since we implement the super-peer architecture in our registry.



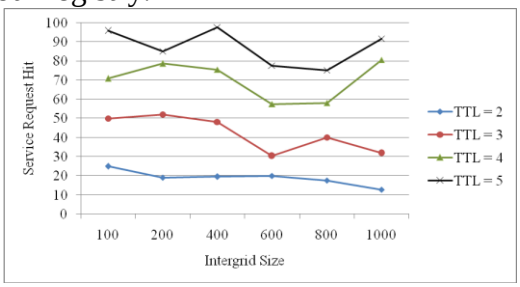


Fig. 6. Discovered Services for generated requests equivalent to 25% of the intergrid size with different TTL values

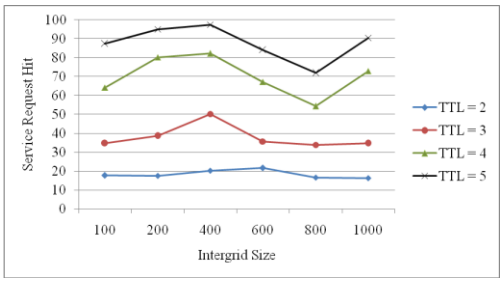


Fig. 7. Discovered Services for generated requests equivalent to 50% of the intergrid size with different TTL values

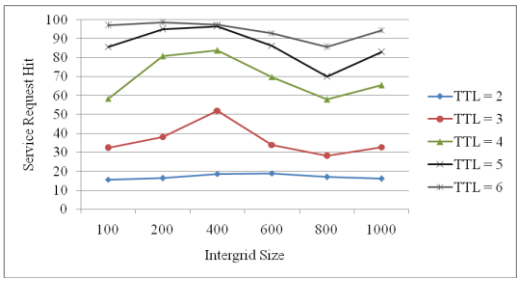


Fig. 8. Discovered Services for generated requests equivalent to 75% of the intergrid size with different TTL values

Fig. 9. Discovered Services for generated requests equivalent to 100% of the intergrid size with different TTL values

It appears from the Fig. 6-9 that the rate of discovered services is low when the TTL is equal to 2. This is because the scope of service request forwarding is limited to within the classes only, or between the heads if it happens that the head node itself generated the request. It is also very clear that the rate of discovered services becomes smaller with the increase of request rate and intergrid size. This can lead to an increase in the overall number of pages.

For example, in Fig. 6 the rate of the discovered services achieves 25% initially, and then drops gradually until 15.62% in Fig. 9. This is because as the service requests increase, the portion of the requests that is sent out of the requester node classes may be higher. This may also happen when the size of the intergrid system is scaled up. Also the four figures unambiguously indicate that the increase of TTL will allow the discovery of more services. For instance, the discovered service rate reaches its highest value 95.83% for an intergrid system consisting of 400 nodes. However, the cases of intergrid size 600 and 800 nodes appear to be different as the rate of discovered services decreases gradually until it reaches the lowest value at size of 800 nodes. The reason behind that is due to the implementation of the load balancing algorithm. In fact, the initial idea of the load balancing mechanism is to split the concept from general to a more specific concept so that we get more classes when a class reaches the maximum predefined size. However, this is hard to be simulated with the simulator as the creation of the nodes, services, and concepts is supposed to be before the intergrid join process starts. Therefore, we simulate the load balancing algorithm by creating new classes during the join process. In this case, if a head of class gets 100 hundred nodes in its class it will reject any new node that wants to join the system. When this happens, the rejected node will create a new class of the same concept and accept other nodes that want join the intergrid and have the same service concepts. Therefore, in the case of intergrid size 600, there are few classes that created, and there are more

in the case of size 800 nodes. So these nodes cannot reach the services that are available beyond the TTL of value 4 for instance. As can be observed in all of the figures the rate of the discovered services starts increasing at TTL of value 5. To further investigate that observation, we increase the TTL value up to 6. As indicated in Fig. 9 the rate of the discovered services is slightly increased at the 800 intergrid size. Meanwhile, it achieves the highest rate with the small intergrid sizes such as 100 and 200 nodes. In addition to that, the rate of discovered services also increases with intergrid size of 1000 nodes. This could possibly be because the created new classes have more number of nodes, which influences the rate of the local discovered services to be higher.

All in all, it is observed that providing more TTL value causes the discovery of more services. However, one may argue that the increase of the TTL may inherit high traffic in the intergrid network. Nevertheless, in our case, the forwarding of service requests takes place only if the request has some semantic relation with the provider, if this not the case then the service request will be forwarded to all neighbors of the head node. Obviously, this will reduce the traffic in the intergrid system and the increase of the TTL value will not cause overhead on the network. Our second point of discussion is on the service request response time of the proposed RD framework. In fact, we use the simulator timer to measure the time between the generation of service request by the requester node until when an answer is given to the requester node. For example, a node may generate a request at time 180000000 (simulation time) and a response may be given at the time of 180017503, therefore the response time

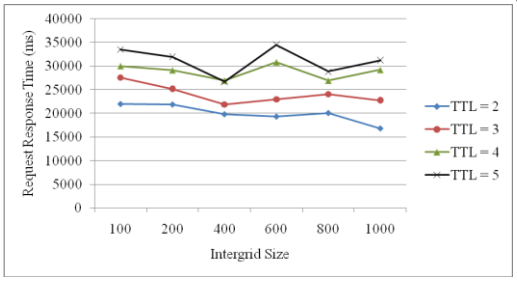


Fig. 10. Service Request Response Time for generated requests equivalent to 25% of the intergrid size with different TTL values

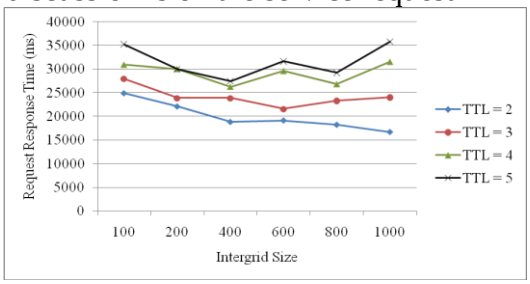


Fig. 6. Service Request Response Time for generated requests that equivalent to 50% of the intergrid size with different TTL values

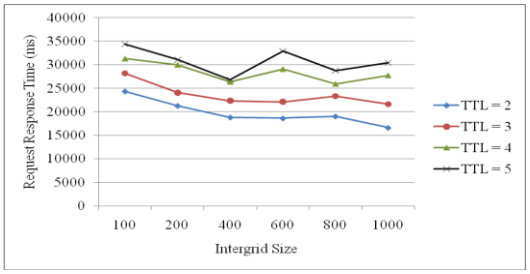


Fig. 7. Service Request Response Time for generated requests equivalent to 75% of the intergrid size with different TTL values

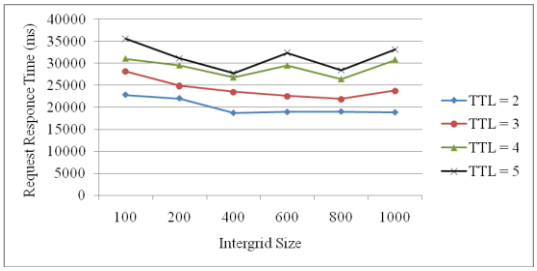


Fig. 8. Service Request Response Time for generated requests equivalent to 100% of the intergrid size with different TTL values

is 17503 millisecond (ms). We calculated the average value of the response time in each set of generated service requests percentage. Fig. 10-13 illustrate these values. It is apparent from all the figures that the increase of service request generation will increase the response time. This also happens when we increase TTL value. For example in Fig. 10, the average response time for generated service request equivalent to 25% of intergrid size of 100 nodes and TTL value 5 is 33486ms. The value becomes considerably higher (35569ms) in Fig. 13 when the service request rate is equivalent to 100% of intergrid nodes. However, the increase of intergrid size does not affect the request response time much, as the curve

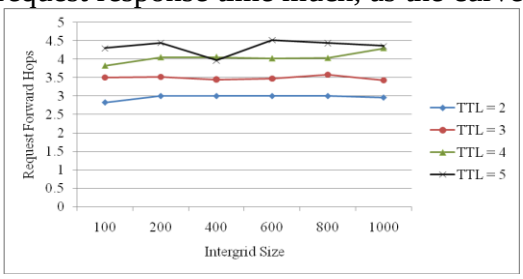


Fig. 14. Average Hops for generated requests equivalent to 25% of the intergrid size with different TTL values

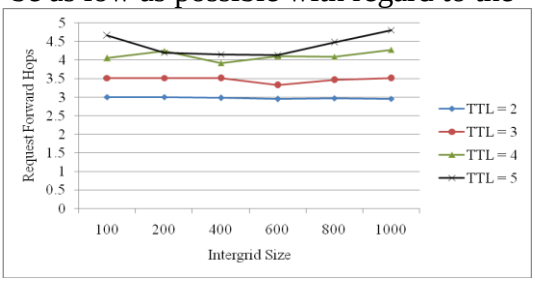


Fig. 9. Average Hops for generated requests equivalent to 50% of the intergrid size with different TTL values

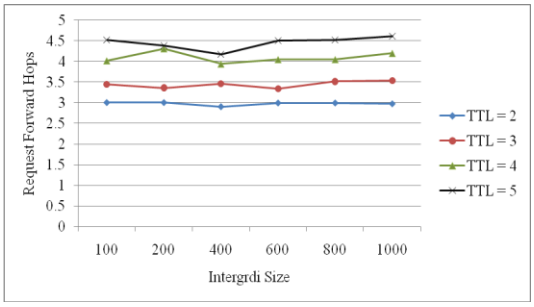


Fig. 10. Average Hops for generated requests equivalent to 75% of the intergrid size with different TTL values

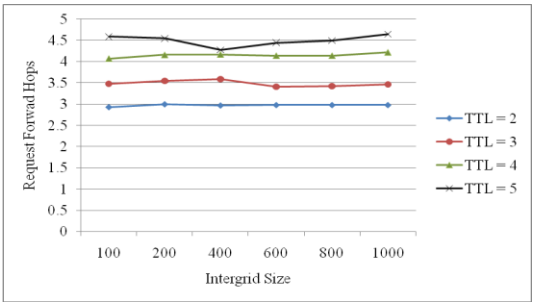


Fig. 11. Average Hops for generated requests equivalent to 100% of the intergrid size with different TTL values

set TTL value. Fig. 14-17 show the average hops of the generated requests. Generally, the average hops values are

of the response time fluctuates in all four figures (10-13). Clearly, this indicates that the increase of the response time is not linearly related to the size of the intergrid nodes. This due to the decentralization of service requests processing as each head processes the service requests that are directed to it only. This ensures that the scale of the intergrid size will not cause performance degradation to the proposed RD system, which ensures sustainability of the system irrespective of the scale of the intergrid users as well as service grids. Another aspect that is much related to the response time is the average number of hops that are crossed during the discovery process, which is supposed to be as low as possible with regard to the

slightly smaller than their respective defined TTL values, regardless of the number of generated service requests.

For example the average hops in TTL 5 has a minimum value of 3.97, as shown in Fig. 14, for the intergrid size of 400 nodes and the request rate is 25%; then it fluctuates to 4.36 as the intergrid size is scaled up to 1000 . However, in all the cases the corresponding rate of the discovered services is good. Therefore, we deduce that having a TTL between 4 to 5 and an intergrid size of 400-1000 nodes will give an acceptable performance to our RD system. A further note on the average hop values when the TTL value is 2 or 3 clearly indicate that the curve of the average hop is quite stable while scoring a poorer service request hits. This happens in the four cases (Fig. 14 – 17). For example, in figure 17 the average hop value for TTL 2 starts with a value of 2.93 and maintains almost the same value to finally end with the value of 2.98 where the size of the intergrid is set to 1000 nodes. Therefore, we can deduce that our RD system can have good performance with TTL values such as 2 or 3 only if the number of concepts is reduced to three or four concepts and the intergrid size is limited to between 100-300 nodes. With this result, it is convincing that the proposed RD system is able to meet the performance requirements for the intergrid RD system. This includes scalability, decentralization and dynamism. The service request hit rates obtained from different intergrid sizes shows that the proposed RD system can scale with the intergrid system as well. The response time has no linear dependency on the scale of the intergrid size which proved the decentralization feature. Lastly, the dynamism feature has been achieved by the fault tolerance mechanism. It worth to mention that, the framework complexity is linear, which renders the system as capable of providing high performance.

4.3 Comparative Study

Since the aim of the study is to provide an advance progress beyond the state-of-art in this field, a comparative study to proof that is therefore needed. Consequently, we compare the proposed RD system with the most promising scalable RDs that we have found in the literature. The most scalable RD systems are the super-peer based RDs [14] and [15] systems, which we have identified as the good candidate for the intergrid level. In fact, our RD system is also an extension of the super-peer model with the addition of the semantic technology into the architecture and optimized discovery algorithm. Therefore, our comparative simulation is done by simulating the same system with and without the use of semantic technology. As such, in order to have a fair comparison between the two situations, we set the intergrid size in the range of 100 – 600 nodes as the stable range where the load balancing mechanism has no much effect on the performance, which will easy the discussion about the scalability of the systems. The random distribution of services to the nodes, the assignment of the number of services in any nodes, and the random generation of the service request for any given node are same in the two situations. The total number of service requests that should be generated by the nodes is equal to their sizes. Fig. 18-20 show the results of the two models in term of service request hit, average response time and average request forward hops.

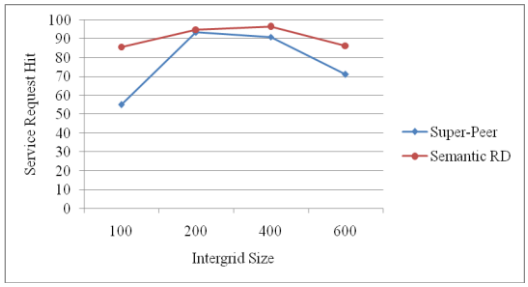


Fig. 18. Discovered Services for generated requests equivalent to 100% of the intergrid obtained with the super-peer model and the semantic RD model

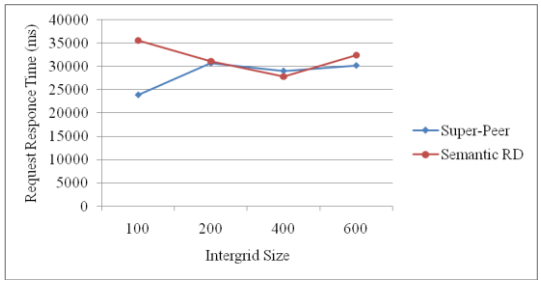


Fig. 12. Average Response Time with the super-peer model and the semantic RD model

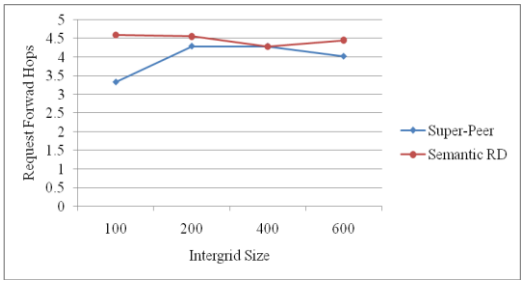


Fig. 20. Average Hops obtained with the super-peer model and the semantic RD model

It is clear from Fig. 18 that the semantic RD system has a better request hit rate compared to the super-peer model in all the intergrid sizes. This is because in super-peer model the services in the classes are not organized in a particular relation, instead they are based on their joining time to the network, which makes it difficult to reach every node in the network. Meanwhile, the average response time of semantic RD model is also slightly higher most of time compared to the super-peer model. This because as the semantic RD model achieves high service request hit rate, it consumes more time. The average number of the hops of semantic is also a bit higher compared the super-peer. This is due to the discovery algorithm of the semantic RD, which optimizes the forwarding of messages in the network so that the service request can reach more nodes while scoring high service request rate. In short, based on the results of the comparative study on the intergrid of 100, 200, 400 and 600 nodes the semantic RD has a better performance than the super-peer model, but we cannot go as far as to generalize these findings because further investigation involving larger intergrid size than what we have used is needed.

5 RELATED WORK

Currently, there is a wealth of work on grid RD (e.g. Globus[4], Condor[5] , [4], [5], and [6]) which can be classified into two classes based on the description component, which are *keyword-based* RD systems and *semantic-based* RD systems. Keyword-based system uses syntactic information and data models such as directories [16] and special

[4] http://www.globus.org/.

[5] http://www.cs.wisc.edu/condor/.

databases to describe and discover the resources and services. Unfortunately, syntactic information and data models are not efficient in describing resources at intergrid level. This is because resources and services are initially described by using multi information services that belong to different grid middlewares. As a matter of fact, much of the efforts in keyword-based RD systems have been focused on achieving the *high performance* requirement; staring from introducing centralized registration models such as Globus MDS-1 [17], R-GMA[6] [18] and Hawakeye [19]; then followed by hierarchical registration models [20], [21] and [22], and lastly peer-to-peer (P2P) registration models [23], [24], [6] and [25]. Keyword-based RD systems that are based on P2P registration models have achieved high performance compared to the centralized and hierarchical models, but we cannot go far as to say that they have achieved full scalability. Moreover, their use of syntactic description, especially at the intergrid level, prevents them from fulfilling the *high searchability* requirement. Semantic-based RD systems, on the other hand, use semantic information and data models (ontology and ontology languages) [9] to describe and discover the resources and services. Although, there is a considerable amount of work on semantic-based RD systems (e.g. [26], [27]), most of the existing approaches fail to achieve *high searchability*. This is due to the lack of a proper use of semantic description mechanism as the semantic technology is initially imported from the semantic web [28]. Actually, we have argued in an earlier study [29] that the main obstacle that leads to the continuous existence of this issue is the ad hoc research nature of these semantic-based RD studies (different research communities doing the same thing by different ways).

[6] Relational Grid Monitoring Architecture: http://www.r-gma.org/index.html

6 CONCLUSIONS

In this paper we presented a new RD framework. The framework has a conceptual model for semantic description that treats the small grids of the intergrid system as services (service grids) and their semantic representation has been based on that; a semantic registry architecture that specifies semantically the distribution of the service grids metadata directories and their management with regard to scalability and dynamism of the service grids metadata; and an agent based discovery algorithm that exploits the description model and the registry architecture to search and select the service grids on behalf of the intergrid user. We have shown the effectiveness of the framework through some discussions and analysis, and an extensive simulation work which has confirmed the effectiveness of the framework.

7 REFERENCES

1. Foster, I., and C. Kesselman The Grid 2: Blueprint for a New Computing Infrastructure. (Morgan Kaufman, San Francisco; 2003).
2. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. & Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 599-616 (2009).

3. Chao-Tung, Y., Wen-Jen, H. & Kuan-Chou, L. in Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing (Springer-Verlag, Taipei, Taiwan; 2009).
4. Lamnitchi, A.L. in Department of Computer Science, Vol. PhD 1-1 (The Univercity of Chicago, Illinois; 2003).
5. Mastroianni, C., D. Talia, and O. Verta A super-peer model for resource discovery services in large-scale Grids. *Future Generation Computer Systems* 21, 1235-1248 (2005).
6. Shen, H. A P2P-based intelligent resource discovery mechanism in Internet-based distributed systems. *Journal of Parallel and Distributed Computing* 69, 197-209 (2009).
7. Subramaniam, R. et al. in GFD-I.145 (Open Grid Forum, 2009).
8. Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43, 907-928 (1995).
9. Chandrasekaran, B., J. R. Josephson, and V. R. Benjamins What are ontologies, and why do we need them? *IEEE Intelligent Systems* 14, 20-26 (1999).
10. Hassan, M.I., and A. Abdulah A New Resource Discovery Framework. *The International Arab Journal of Information Technology (IAJIT)* 8 20-28 (2011).
11. Kovacevic, A., Kaune, S., Mukherjee, P., Liebau, N. & Steinmetz, R. Benchmarking Platform for Peer-to-Peer Systems , vol. 46, no. 3, 2007. . *it - Information Technology (Methods and Applications of Informatics and Information Technology)* 49, 312-319 (2007).
12. Mastroianni, C., D. Talia, and O. Verta Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models. *Parallel Computing* 34, 593-611 (2008).
13. Yatin, C., Sylvia, R., Lee, B., Nick, L. & Scott, S. in Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (ACM, Karlsruhe, Germany; 2003).
14. Mastroianni, C., Talia, D. & Verta, O. A super-peer model for resource discovery services in large-scale Grids. *Future Generation Computer Systems* 21, 1235-1248 (2005).
15. Mastroianni, C., Talia, D. & Verta, O. Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models. *Parallel Computing* 34, 593-611 (2008).
16. Tuttle, S., A. Ehlenberger, R. Gorthi, J. Leiserson, R. Macbeth, N. Owen, S. Ranahandola, M. Storrs, C. Yang Understanding LDAP Design and Implementation. (IBM, 2004).
17. Fitzgerald, S., I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke in 6th IEEE Symposium on High Performance Distributed Computing 365–375 (IEEE Computer Society Press, 1997).
18. Cooke, A. et al. in On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Vol. 2888 462-481 (Springer Berlin / Heidelberg, 2003).
19. Zanikolas, S. & Sakellariou, R. A taxonomy of grid monitoring systems. *Future Generation Computer Systems* 21, 163-188 (2005).
20. Steven, F. in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (IEEE Computer Society, 2001).
21. Schopf, J.M. et al. Monitoring the grid with the Globus Toolkit MDS4. *Journal of Physics: Conference Series* 46, 521 (2006).
22. Ruay-Shiung, C. & Min-Shuo, H. A resource discovery tree using bitmap for grids. *Future Generation Computer Systems* 26, 29-37 (2010).
23. Trunfioa, P., D. Taliaa, H. Papadakisb, P. Fragopouloub, M. Mordacchinic, M. Pennanend, K. Popove, V. Vlassovf, and S. Haridi Peer-to-Peer resource discovery in Grids: Models and systems. *Future Generation Computer Systems* 23, 864–878 (2007).
24. Marzolla, M., Mordacchini, M. & Orlando, S. Peer-to-peer systems for discovering resources in a dynamic grid. *Parallel Computing* 33, 339-358 (2007).
25. Brocco, A., Malatras, A. & Hirsbrunner, B. Enabling efficient information discovery in a self-structured grid. *Future Generation Computer Systems* 26, 838-846 (2010).
26. Ludwig, S.A., and S. M. S. Reyhani Introduction of semantic matchmaking to grid computing. *Journal of Parallel and Distributed Computing* 65, 1533-1541 (2005).

27. Said, M.P., and I Kojima S-MDS: Semantic Monitoring and Discovery System. *Journal of Grid Computing* 7, 205-224 (2009).
28. Berners-Lee, T., Hendler, J. & Lassila, O. The Semantic Web. *Scientific American* 284, 34-43 (2001).
29. Hassan, M.I. & Abdullah, A. in 2010 International Symposium in Information Technology (ITSim), Vol. 3 1286-1296 (IEEE Xplore, 2010).