A Resource Discovery Framework for Semantic Grids Based on the Interface-Based Modular Ontology Formalism

Faezeh Ensan *, Alireza Ensan †, Weichang Du *

* *Faculty of Computer Science, University Of New Brunswick*
Fredericton, NB, Canada
† *Faculty of Electrical and Computer Engineering, Ferdowsi University Of Mashhad*
Mashhad, Iran
* {faezeh.ensan,wdu}@unb.ca, † al_en52@stu-mail.um.ac.ir

Abstract

Semantic grids refer to those grids that their resources and services have been described by the means of semantic meta data and ontologies. In this paper we propose a resource discovery framework for semantic grids using the notion of modular ontologies. We exploit interface-based modular ontology formalism whose through ontologies can be described and be accessed by a set of interfaces. We show how this formalism help looking for distributed resources in semantic grids. We describe the architecture of the resource managers nodes and their resource discovery algorithms.

1 Introduction

Grid has arisen as an emerging technology in resent years which enables sharing computational resources between different systems that are distributed across virtual organizations. Grid technology mostly comes with a middleware which is responsible for resource management. This layer provides services for discovering and allocating available resources to consumers considering administrative pollices, allocation rules and constrains and quality of service [1]. Resource discovery is one of the most important services provided by the resource management middleware which given a query, attempts to find the most appropriate resources fulfilling the query requirements from available resource providers.

Resource discovery can be more semantically accurate if available resource are described using semantic Web techniques and specially ontologies. Using ontologies, resource providers and consumers can query and share their resources with more flexibility without employing limiting symmetric attribute-based techniques [2]. The idea of applying semantic Web achievements in the computational grids area leads to introduction of the notion of semantic grid [3]. In semantic grids, computational resources and services are described by means of semantic data models and meta data.

Recently several researches [2, 4, 5] are performed in order to facilitate the resource management tasks by defining a few global or local ontologies for grid resources. Developing a global comprehensive ontology which describes all available resources among distributed machines is a problematical task. Some experts should recognize all of concepts related to the all potential resource. This ontology may be subject of frequently change because of introduction of new resources or extracting new relationships between them. In addition, within grid environment, resource holders are independent, self contained virtual organizations and it is hardly successful to provide such a comprehensive ontology such that be accepted by all of them and all of new incoming ones. Few existing researches which address these problem by defining multiple local ontologies for every node in grid are still in initiate stages of maturity and should deal with ontology matching and merging challenges.

In this paper we propose a new framework for ontology-driven resource discovery in grids, utilizing the notion of modular ontologies. Modularization is a new alternative ontology development process against the traditional monolithic approach. In this paper we employ the *interface-based* modular ontology formalism that we introduced in [6, 7]. Using this formalism, our resource discovery framework provides different grid nodes with the capability to define and use their own ontologies while these ontologies can have connections to other nodes' ontologies and can be easily communicate.

978-0-7695-3401-5/08 $25.00 © 2008 IEEE
DOI 10.1109/SKG.2008.73

2 Resource Discovery Framework

In this section we discuss the grid resource discovery framework based on the interface-based modular ontologies.

The *interface-based* modular ontology formalism is a new designed formalism for creating modular ontologies. Through this formalism, a modular ontology is defined as a set of ontology modules and interfaces where an interface is a set of concept and role names and their inclusion axioms and a module is an ontology in any description logic language. An ontology module can either *realize* or *utilize* an interface. A realizer module should provide definitions and assertions for roles and concepts of a specific interface. On the other hand, a utilizer module augment its knowledge base with the definitions and assertions provided by the other (realizers) modules. In the definition of modular ontologies, there exist also a configuration function which specifies which utilizer and realizer modules should be connected.

Each grid includes nodes which provide resources (providers), consume resources (consumers) or manage resources (resource mangers). Each grid node has its own local ontology module which describe its available resources. These ontologies may describe different concepts or may give different categorizations and axioms about the same resources. Interface-Based modular ontology formalism is employed to integrate these distributed ontologies and extract the most appropriate meaning for the terms of resource queries posed to different heterogenous nodes. In following, we first scrutinize the resource advertisement process performed by provider resource. Secondly, we describe the architecture of resource managers and go through the detail of task each of the components of the architecture should perform.

2.1 Resource Advertisement

As we mentioned in the previous paragraphs, every provider node has its local ontology that describe its available resources. With the purpose of joining a grid, a provider node should define an ontology interface for its ontology or select to *utilize* or *realize* among existing interfaces of other nodes. Figure 1 shows an example of four resource providers and a resource manager node with different ontologies and interfaces. This example illustrates the main different ways that a provider can join a grid:

1 A provider node can introduce a new interface ontology and insert it to the interface pool of the resource manager node. As an explanatory example, consider that the first resource provider (p1) has a set of available resources and a simple ontology with the

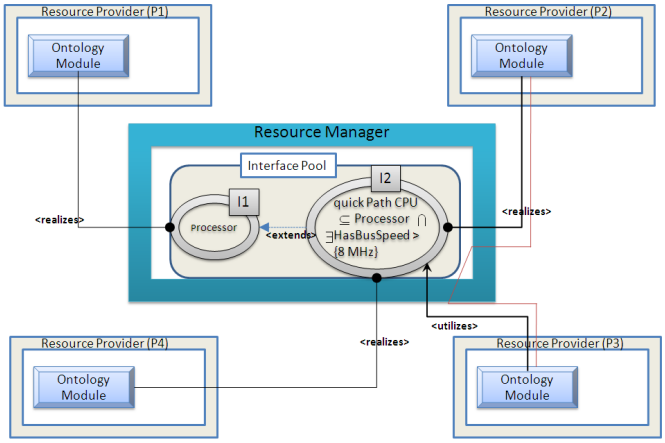

Figure 1. An example of a grid with different nodes and distributed ontology modules.

following ABox assertions:

```
Processor(Intel Celeron processor 560)
Processor(Intel
 Pentium 4 processor 532)
Processor(Motorola MC68EC060RC75)
```

According to Figure 1, this node has inserted a new interface (*I1*) to the resource manager interface pool and has set itself as a realizer ontology module for it.

2 Alternatively, a provider node may choose one of the existing interfaces in interface pool and introduce an extension to it by defining new concepts, roles or axioms. It should insert this extending interface to the resource manager pool as well. In the example shown in Figure 1, the second provider node (p2), has introduced interface *I2* as an extension to the existing interface *I1* and specified itself as a realizer module for this interface. *I2* gives new information about processors and specify that verb"fast path CPU" is a sub set of those processors that have a bus speed greater than 8 MHz. Suppose the ABox of the local ontology module of p2 be as follows:

```
Processor(Motorola 68000 Macintosh)
Processor(Motorola PowerBook 100)
HasBusSpeed(Motorola
68000 Macintosh, 8 MHz)
HasBusSpeed
(Motorola PowerBook 100, 16 MHz)
```

According to the above knowledge base, we can rec-

ognize that p2 gives new information about processors independently of the knowledge provided by p1.

3 The third option would be the provider node decides to use the ontologies provided by other nodes for describing its available resources. In this case it should select among existing interfaces from the resource manager interface pool and also select an appropriate realizer module for the interface. In figure 1, p3 has selected *I2* and also selected p2 as the realizer module in order to attain a definition for processors and fast processors.

4 The last possibility is that a provider node find a suitable interface in the interface pool which can appropriately describe its resources but decides to uses its own ontology as the realizer of that interface inserted of using existing available ontology modules from other provider nodes. For instance in Figure 1, p4 has a few processors and fast processors which can be described using the interface *I2*. However it has prefered not to use the ontology provided by already existing realizer module p2. Instead it introduces itself as an new realizer module for *I2* with following TBox:

$$\texttt{Intel} \sqsubseteq \texttt{Processor}$$
$$\texttt{Fast Processor} \equiv \texttt{Intel}$$

From the point of view of the p4 node, all of fast processors are of type Intel. Nonetheless, this interpretation has not affect on the ontology of p2. It also has not affect on the knowledge of p3 about processors unless it has change the configuration function and set its realizer module equal to p4.

2.2 Resource Manager Architecture

Figure 2 shows the architecture of the resource managers. The main components of the resource manager are the interface pool, resource matching unit, node withdrawal unit and description logic reasoner. Interface pool is a knowledge warehouse where grid nodes register their interface ontologies (as we saw in the previous section). It has mechanisms for inserting, deleting, retrieving and searching among ontology interfaces as well as the name and address of the grid nodes connected to interfaces.

Through our proposed framework for grid resource discovery, resource requests are expressed as RDQL query expressions using concept names and roles defined in the knowledge base of interfaces that are present in the interface pool. For instance, in the sample grid shown in Figure 1, a consumer node may pose a resource query for all processors which has a bus speed more that 8 MHz as follows: $\texttt{(Processor)} \cap \exists \texttt{HasBusSpeed} > 8Mhz$

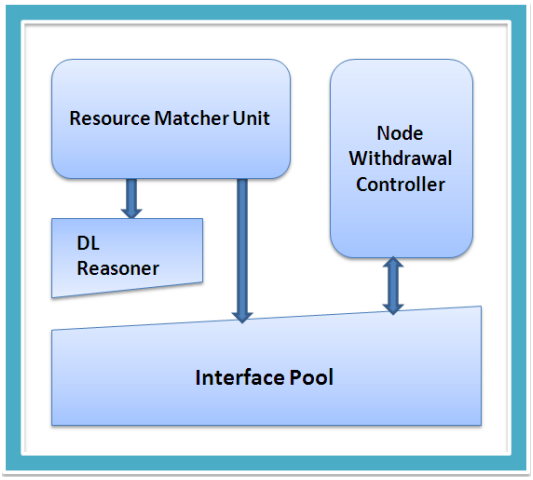

Figure 2. Resource Manager Architecture

The resource manager node needs a reasoner component to process the available ontologies and produce suitable result set for a description logic resource query. In following, we talk about the two other components in more detail.

Resource Matching Unit: It receives The RDQL query expressions from resource consumers. Subsequently, it looks for those interfaces that have all the concept names and role name of the received query in the interface pool and redirect the query to the related modular ontologies. For example consider the $\texttt{(Processor)} \cap \exists \texttt{HasBusSpeed} > 8Mhz$ has been posed in the resource manger discovery unit in the grid illustrated in Figure 1. The unit will find interface *I2* that includes $\texttt{(Processor)}$ concept name and $\texttt{HasBusSpeed}$ role name. Using the interface pool, resource discovery unit find that the p2,p3 and p4 nodes are connected as the *I2*, consequently, it redirect the query to their knowledge base and examine the return result set. The result set of p2 is equal to $\{\texttt{Motorola 68000 Macintosh Motorola PowerBook 100}\}$. The result set of p3 is an empty set for the reason that the only individual of the p3 knowledge base $(\texttt{Intel Celeron processor 560})$ is not a fast processor and therefore its bus speed is less than 8 MHz. p4 does not have any ABox assertions so its result set is also empty. In circumstances where there are more than a node with non empty result sets to the resource query, the resource manager should employ quality of service measures and resource allocation policies that are beyond the scope of this paper.

Node Withdrawal Controller: When a resource provider node is leaving a grid, it should send a withdrawal message to the resource manager node. Node withdrawal controller receives these messages and update the interface pool and also sends a couple of update message to the dependent grid nodes. The withdrawal of those nodes that

Algorithm 1 Node Withdrawal Control

Input: A node withdrawal message
$msg = withdraw\langle \quad px, G\rangle$
where px is a provider node in the gird G.

RI := (Interface Pool).getAllInterfacesRealizedBy(px)
if (RI.size$>$ 0)
  for all $I \in$ RI
    $UtilizerNodes$:= (Interface Pool).getAllNodeNamesUsing(I)
    $RealizerNodes$:=
    (InterfacePool).getAllNodeNamesRealizingOtharThan(I,px)

    if (UtilizerNodes.size$>$ 0)
      if (RealizerNodes.size$>$ 0)
        Send update configuration message
        to(UtilizerNodes, RealizerNodes)
      else
        Send exit message to(UtilizerNodes)
        Delete I
    else
      if (RealizerNodes.size$=$ 0)
        Delete I

realize a couple of ontologies will affect the grid interface pool and also may affect on other nodes. If a node is the only node which realizes an ontology that is used by another node, its withdrawal makes the user nodes do not have the meaning for their resource, subsequently they also should leave the grid. However if the resource manager find that there are other nodes that can help the utilizer modules describe their resource after laving a specific node, it let the utilizer node informed to change its configuration and use other available ontologies. Algorithm 1 demonstrates the main body work of this unit.

As an explanatory example for the algorithm, Let's study the withdrawal process of the grid nodes p2 and p4 of the gird which is demonstrated in Figure 1. p2 has a realizer ontology module for the interface $I2$ (RI.size$>$ 0) and $I2$ has been used by another ontology from p3 ($UtilizerNodes.size > 0$). In addition there exist p4 which realizes $I2$ ($RealizerNodes.size > 0$), consequently the withdrawal controller send an update message to p3, and get it informed that the p2 node does not any more exist and it can change its realizer module from p2 to p3 ($Sendupdateconfigurationmessage$). Now consider that the p4 node also intend to leave the grid. p4 has a realizer ontology module for the interface $I2$ (RI.size$>$ 0) and $I2$ has been used by p3 ($UtilizerNodes.size > 0$). Since p2 has been already left the grid, there are no In addition no node that realizes the necessary ontology of p3. Consequently, the withdrawal controller send an exit message to p3 ($Sendexitmessage$) and ask it to leave the grid. Furthermore $I2$ will be removed from interface pool.

3 Conclusion

In this paper, we presented a modular ontology based resource discovery framework for semantic grids. The proposed resource discovery framework in this paper has the following contributions regarding to the existing solutions in the literature:

- Each grid can have its own ontologies. The interface-based modularity formalism supports the distribution of these ontologies and their communications
- Using interface ontologies and contrary to most existing peer to peer approach, the proposed framework does not required a huge amount of message passing between all of grid nodes in order to find the appropriate resources for a specific query. Using interface pool component, the resource manager can easily find the suitable candidate for resource requests.
- The withdrawal process is efficient in the light of this face that when a node leave a grid, it only affects those node that uses its knowledge about its resources and not other grid nodes.

References

[1] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Softw., Pract. Exper.*, vol. 32, no. 2, pp. 135–164, 2002.

[2] H. Tangmunarunkit, S. Decker, and C. Kesselman, "Ontology-based resource matching in the grid - the grid meets the semantic web," in *International Semantic Web Conference*, 2003, pp. 706–721.

[3] D. de Roure, N. R. Jennings, and N. Shadbolt, "The semantic grid: A future e-science infrastructure," pp. 437–470, 2003.

[4] M. Siddiqui, T. Fahringer, J. Hofer, and I. Toma, "Grid resource ontologies and asymmetric resource-correlation." in *NODe/GSEM*, ser. LNI, R. Hirschfeld, R. Kowalczyk, A. Polze, and M. Weske, Eds., vol. 69. GI, 2005, pp. 205–219.

[5] H. Jin, Y. Pan, N. Xiao, and J. Sun, Eds., *Grid and Cooperative Computing - GCC 2004: Third International Conference, Wuhan, China, October 21-24, 2004. Proceedings*, ser. Lecture Notes in Computer Science, vol. 3251. Springer, 2004.

[6] F. Ensan and W. Du, "Aspects of inconsistency resolution in modular ontologies," in *Canadian Conference on AI*, 2008, pp. 84–95.

[7] ——, "An interface-based ontology modularization framework for knowledge encapsulation," in *International Semantic Web Conference*, 2008.