

OWL-SR: Unified Semantic Service and Resource Discovery for Grids

Frederick Lee¹, Sukesh Garg¹, Shishir Garg¹

¹ France Telecom R&D, 801 Gateway Boulevard, Suite 500, South San Francisco, CA, USA
{ frederick.lee, sukesh.garg, shishir.garg }@orange-ftgroup.com

Abstract. Service discovery has been an age old issue for grids. Coupled with it is the need for resource discovery. Together they form the building block of a grid computing framework. However it is not sufficient to just have plain vanilla resource and service discovery. There is a clear need to expose semantic rich information associated with resources to build intelligent services. The semantic descriptions of these resources have an associated identity and behavior. Furthermore, it is important to define mechanisms for their creation, management and protocols for processing, exchange and customization. There is no reference architecture or systematic framework in place for designing semantic components and applications. We aim to capture the knowledge and vocabulary for service and resource discovery by capturing them in ontologies for service and resource discovery. We develop a lightweight framework extending CMU's matchmaker to introduce semantics for resources. The paper also leverages the mechanism presented in [8] to develop a unified semantic API for both services and resources.

Keywords: UDDI, MDS, Semantic, Discovery, GRID

1 Introduction

Service discovery has been an age old issue for grids. Coupled with it is the need for resource discovery. Together they form the building block of a grid computing framework. However it is not sufficient to just have plain vanilla resource and service discovery. There is a clear need to expose semantic rich information associated with resources to build intelligent services. These semantic descriptions have an associated identity and behavior. Furthermore, it is important to define mechanisms for their creation, management and protocols for processing, exchange and customization. There is no reference architecture or systematic framework in place for designing semantic components and applications. We aim to capture the knowledge and vocabulary for service and resource discovery by capturing them in ontologies for service and resource discovery. We develop a lightweight framework extending CMU's matchmaker to introduce semantics for resources. The paper also leverages the mechanism presented in [8] to develop a unified semantic API for both services and resources.

The web services communities have addressed the need for service discovery, before grids were anticipated, via an industry standard called UDDI. Corporations have already embraced and deployed private UDDI registries for use within their organizations and have registered numerous services within these registries. There is tremendous interest in the grid community to adopt the web services framework for providing the essential functions of a grid such as service/resource discovery and registration. The most prominent specification thus far is the web services version of the Monitoring and Discovery Service (WS MDS), also known as the MDS4 component of the Globus Toolkit version 4 (GT4). However, MDS is more suited towards resource registration and discovery. In [8] the authors presented a Mediator service that offers a single API for registering and discovering both resources and services. The API is robust enough to allow for a unified query for both services and resources on which the services are running. It allows for querying a resource with a given resource property in a specific range and at the same time ensuring that the requested service is available. This converged approach considerably reduces the result set by eliminating services which are out of scope or resources that are not available. An example of such a query is "Find a Translation Service on a Resource where CPU utilization is less than 40%". The mediator service leverages the UDDI registry and the MDS registry to fulfill the request. It uses the UDDI registry for service discovery and the MDS directory for resource discovery. The unified API leverages the best in class APIs while hiding the complexity from the user. However, the service discovery API that was presented did not permit the capturing of semantic information associated with both the service and the resource or the ability to discover services and resources based on the associated semantic information.

This paper presents OWL-SR as a Mediator service that offers a single API for registering and discovering both resources and services leveraging semantics. The API is robust enough to allow for a unified query for both services and resources on which the services are running. It allows for semantically querying a resource with a given resource property in a specific range and at the same time ensuring that the requested service is available. It uses the CMU Matchmaker registry for semantic service discovery and the MDS directory for resource discovery. The unified API leverages the best in class APIs while hiding the complexity from the user. The paper further leverages the Common Information Model (CIM) to create ontology for resources.

2 Background

2.1 Service Discovery Requirements

Service Discovery is a critical component required to deliver on the promise of truly loosely coupled Service Oriented Architecture where applications can dynamically discover and use available services on the web. However, typically these sort of service discoveries need to not only identify available services from a feature and

capability perspective but also the underlying performance and availability of the resources on which these services are running and therefore impacting the overall complete performance of the service and resulting composite application. Because applications and resources are distributed globally and across different virtual organizations that are inherently heterogeneous, Service Discovery becomes extremely important. The current approaches to service/resource discovery such as UDDI or MDS for that matter do not have sufficient expressiveness and efficient matchmaking abilities in their query language. A relational query language would enable more expressiveness but would be difficult to implement in scale.

In order to determine resources and services, users need to not only use resource functionality information in finding the right service but also need to apply their own selection policies with regards to non-functional characteristics of services such as reliability, invocation cost, provenance, quality of service, reputation, etc. A service discovery language needs to be expressive enough to capture this metadata. Furthermore, this needs to be amenable to automatic processing. [1]

Ludwig and Reyahi [2] identify the requirements for a grid service discovery framework to be as follows:

- High degree of flexibility and expressiveness – different advertisers would want to describe services with varying degrees of complexity and completeness. The tools have to support both in depth description and complexity and yet permit other aspects to be not clearly specified
- Support for subsumption – matching should not be restricted to simple service name comparison – more complex matching based on relationships specified in a type system needs to be supported
- Support for data types – attributes such as quantities should be part of the service description. This is best supported through data types
- Matching process should be efficient – it should not cause excessive delays for the requester that would prevent effectiveness
- Appropriate syntax for Grids – format should be compatible with grid and web technologies
- Flexible and modular structure – that allows grid applications to describe their context semantics and grid services to describe their service semantics in a modular manner
- Lookup for matched services – mechanism for lookup and invocation of matched services

2.2 Globus Toolkit Monitoring and Discovery Service

The Monitoring and Discovery Service (MDS) from Globus Toolkit [5] is a set of web services for monitoring and discovering resources in a grid. A query typically returns a set of resources that can complete a job in the shortest amount of time with the most efficient use of resources. The resource selection process involves using static information such as CPUs, clock speed, physical memory, virtual memory and

disk space. In addition, dynamic information is also used such as current available CPUs, number of jobs queued and current CPU and memory utilization. The MDS service for gathering the resource information is the Index Service. It collects the information and publishes it as a resource property and provides it via a web service. At any given time, the information presents the current configuration and state of the grid resources. It is maintained as a set of Web Service Resource Framework (WSRF) resource properties. MDS also provides subscription and notification on a resource property using the Trigger Service. It can execute policies, defined earlier, once a specified threshold is exceeded. The Index service and the Trigger services are two critical services built on the MDS Aggregation Framework. The framework offers a hierarchical index service registration enabling the consolidation of data from multiple locations into a single point. Queries can be made against the resource attributes which represent the state and configuration of the resource. The resource attributes are maintained in a “soft state” by associating a lifetime to them. This ensures the removal of stale information. Since MDS maintains configurations and services in a soft state, constant updates are required in order to renew/update these services. Services are registered using the service registration file. The service registration file contains the grid resource, the service group the resource should register with and service configuration parameters.

MDS does solve the resource discovery and monitoring problem. However, MDS does not offer the flexibility and functionality offered by UDDI for service discovery. For example, there is no taxonomy available in MDS such as UDDI’s “category bag”. Furthermore, there is no simple way to make a query for a specific service where the resource attribute falls in a specified range. MDS needs complex XPath query language in order to identify specific service. Leveraging UDDI to provide service information and MDS to provide resource information and coupling them under a single API can improve service/resource discovery and registration. [8] attempts to solve the problem but lacks in not being a semantic approach.

2.3 UDDI

Beyond grid computing, the problem of service discovery needs to be addressed more generally in the web services community. Flexibility is the key factor since users are looking for specific services among the millions of unspecified services. Although there are different ways of doing this, the web services standards committees address this requirement through a specification called UDDI (Universal Description, Discovery, and Integration). A UDDI registry enables a business to enter three types of information in a UDDI registry – white pages, yellow pages and green pages. UDDI’s intent is to function as a registry for services just as the yellow pages is a registry for businesses. Just like in Yellow pages, companies register themselves and their services under different categories. In UDDI, White Pages are a listing of the business entities. Green pages represent the technical information that is necessary to invoke a given service. Thus, by browsing a UDDI registry, a developer should be able to locate a service and a company and find out how to invoke the service. Furthermore, UDDI has been designed to support not just developers or users but also

applications that can dynamically in real time search for and consume services programmatically with no manual intervention. It is therefore considered a key enabler for service composition.

UDDI offers APIs for publishing and inquiry of web services. These APIs are quite simple to use as the UDDI registry itself is exposed as a web service using SOAP. In addition, UDDI's tModel is a general purpose data structure of linking metadata outside of UDDI. The tModel is used for example for defining service types or for pointing to metadata about a business and service registered with UDDI. Furthermore, since UDDI is expected to hold hundreds of thousands of business entities and services, it is necessary to categorize them for easy use based on, for example, geography, industry type, service type, etc. UDDI offers a useful concept called a Category Bag to create its taxonomy.

However, UDDI also has a number of shortcomings. The readers can refer to [3] for a paper that discusses the shortcomings of UDD. Improvement of the UDDI standard is continuing in full force and UDDI version 3 (V3) [4] was approved as an OASIS Standard. However, UDDI today still has issues that have not been addressed. One key failing in UDDI that prevents its use today for Grid Computing is that UDDI is not well suited for registering and management of stateful resources.

2.4 CMU Semantic MatchMaker

The CMU Semantic Matchmaker is an entity that allows web services to locate other services, provide a solution to the problem of matching, and allow for full implementation of interoperate service providers on the web. It introduces OWL-S, a OWL-based language for describing service capabilities. It shows how semantic matching between advertisements and requests is performed. It focuses on the problem of locating web services on the basis of the capabilities that they provide. Their solution to this problem used a language to express the capabilities of services, and the specification of a matching algorithm between service advertisements and service requests, one that recognizes when a request matches an advertisement. For this they adopt OWL-S as a service description language, because it provides a semantically-based view of web services, including the abstract description of the capabilities of the service, the specification of the service interaction protocol, and the actual messages that it exchanges with other web services.

Standards such as SOAP and WSDL are designed to provide descriptions of message transport mechanisms, and for describing the interface used by each service. However, neither SOAP nor WSDL are of any use for providing the automatic location of web services on the basis of their *capabilities*. Another emerging XML based standard is UDDI. It provides a registry of businesses and web services. UDDI describes businesses by their physical attributes such as name, address and the services that they provide. In addition, UDDI descriptions are augmented by a set of attributes, called TModels, which describe additional features such as the classification of services within taxonomies such as NAICS. But because UDDI does not represent service

capabilities, it is of no use for locating services on the basis of what they provide. Through the tight connection with OWL+OIL, OWL-S supports the need for semantic representation of services. Furthermore, OWL+OIL allows for the definition of relations between concepts. The main limitation of OWL+OIL is its lack of a definition of rules and an associated reasoner. Therefore, they coupled OWL-S with RuleML. RuleML can describe constraints related to input and output, and also preconditions and effects for planning.

The CMU Semantic Matchmaker is also a web service that helps make connections between service requesters and service providers. The Matchmaker serves as a "yellow pages" of service capabilities. The Matchmaker allows users and/or software agents to find each other by providing a mechanism for registering service capabilities. Registration information is stored as advertisements. When the Matchmaker agent receives a query from a user or another software agent, it searches its dynamic database of advertisements for agents that can fulfill the incoming request(s). Thus, the Matchmaker also serves as a liaison between a service requester and a service provider.

The CMU OWL-S Matchmaker employs techniques from information retrieval, AI, and software engineering to compute the syntactical and semantic similarity among service capability descriptions. The matching engine of the matchmaking system contains five different filters for namespace comparison, word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching. The user configures these filters to achieve the desired tradeoff between performance and matching quality.

2.5 Common Information Model

The Common Information Model (CIM)[9] is conceptual information model for describing computing and business entities in internet, enterprise and service provider environments. It provides a consistent definition and structure of data, using object oriented techniques. The CIM includes expressions for common elements that must be clearly presented to management applications like object classes, properties, methods and associations to name a few. CIM uses a set of terminology specific to the model and the principles of object oriented programming.

The Common Information Model (CIM) provides a method for describing and interacting with many different kinds of resources in an heterogeneous IT environment. Using CIM, systems management applications have a single model for communicating with these different IT resources. We aim to use the CIM model to build the ontology for resources.

3 Architecture

The DAML-S/UDDI Matchmaker augments UDDI registries with an additional semantic layer for capacity match making. It leverages DAML ontologies. Services that are advertised using DAML-S are also available inside the UDDI registry.

The Globus Toolkit's MDS provides a resource registration and discovery service. The main function of MDS is to provide a mechanism to query and update resource information. MDS is an ideal way to represent dynamic attributes of a resource but its information model is too simplified to show variable aspects of the relationship.

We marry the DAML-S/UDDI Matchmaker to Globus Toolkit's MDS using the Unified Mediator Engine (UME) [8]. We have named this UME engine OWL-SR. The OWL-Semantic Resources (OWL-SR) UME is built on the Unified Mediator Architecture (UMA). The UMA converge both DAML-S/UDDI Matchmaker and MDS with both solutions working in a symbiotic relationship. Services are registered in DAML-S/UDDI matchmaker and resources are registered in MDS. The UMA uses the OWL-SR UME to present services and resources in a distributed system using semantic API.

Figure 1 highlights the architecture of the Information Mediator based on the DAML-S/UDDI Matchmaker and the Globus MDS framework.

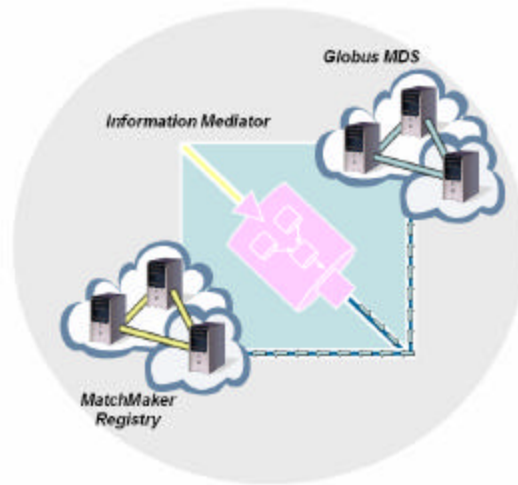


Fig. 1. OWL-SR Unified Mediator Architecture (UMA)

The OWL-SR Information Mediator mediates the communications between client and backend registries such as Matchmaker and MDS. The OWL-SR Information

Mediator communicates with the Matchmaker and the MDS registries using different protocols.

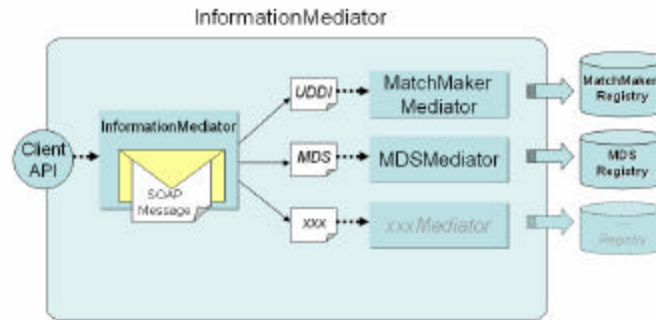


Fig. 2. Information Mediator

This pluggable framework guarantees maximum flexibility to enable the addition of an auxiliary specification to the existing mediator framework without any major modification.

With the abstraction layer for a heterogeneous registry environment, the client is agnostic to whether the service entry is for the Matchmaker or the MDS registry. The OWL-SR Information Mediator will transform any messages passing through it to the appropriate protocol understood by the respective registry.

An important aspect of the Semantic API is the ability to semantically match the properties to the resources. To enable this we present the ontology for the resources.

4 Service and Resource Discovery Ontology

The current implementation of CMU MatchMaker doesn't contain ontology for resources since it only provides semantic service discovery. We developed resource ontology so the OWL-SR can provide unified semantic discovery API to the client.

There is an ongoing effort to define metadata information model on distributed computing system resources called Common Information Model (CIM) led by Distributed Management Task Force (DMTF). DMTF is allied with GGF to introduce CIM into GRID OGSA scheme. We used CIM to describe ontology of GRID resource so that we can keep consistent compatibility to the possible release of the next generation GGF OGSA. CIM provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM's common definitions enable us to easily define semantically rich metadata information for the GRID resource.

Figure 3 shows the ontology for the resource profile which is used to query a resource. By using resource profile, client can compose a single discovery message which is compliant to the OWL-S then complete the entire discovery within a single transaction.

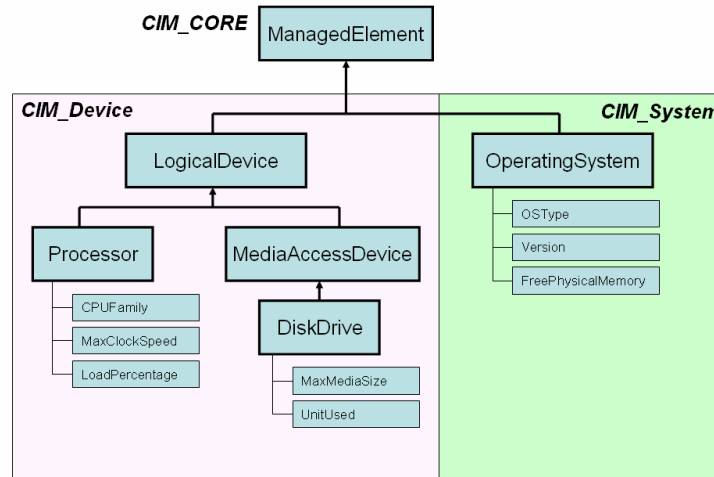


Fig. 3. Resource Ontology

The resource ontology contains the concepts of the resource domain specified by CIM classes and properties. And these characterize the resource for the advertisement, discovery and matchmaking. Information Mediator can access the appropriate properties on class by following logical ontology tree.

4.1 Sequence of Operations

In this section, we present the sequence diagrams for two main scenarios – adding a new service to a new resource and querying for service with resource properties. Other operations like updating, deleting a service/resource, etc. are similar or subsets of the above mentioned scenarios and are omitted for the sake of brevity.

Add a new Service with a new Resource

Figure 4 shows how a client can add the service with an existing resource in the MDS registry. The service registration request leverages the existing resource entry and dynamically establishes the relationship between the service and the resource.

The client sends a service registration request to the Information Mediator with the existing resource information specified in the endpoint for the service. The Information Mediator analyzes the message signature and retrieves the information

about the resource that the client wants to use. The Information Mediator forwards the service registration request to the MatchMaker registry and waits for completion. Upon receiving the registration response from the MatchMaker registry, the Information Mediator creates an association between the new service and the existing resource by sending the new service key and service name to the resource.

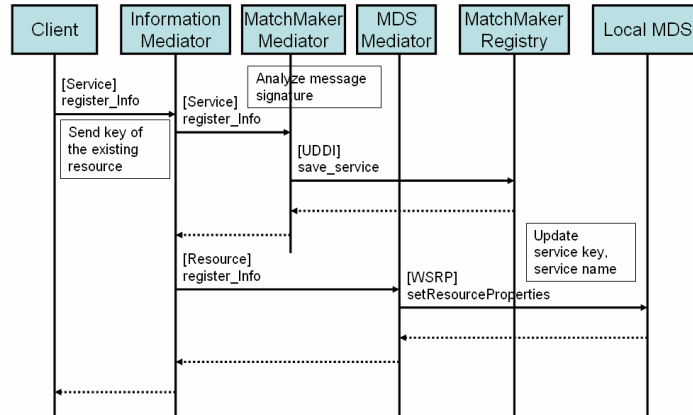


Fig. 4. Sequence Diagram – Add new service with existing Resource

Query a Service

Figure 5 shows how a client can query the service with a given set of resource properties. Every service has at least one resource associated with it. The operation returns the set of services that satisfy the query parameters which exist on resources which satisfy the resource query parameters. This operation is transparent to the client. The client sends the service query to the Information Mediator. The Information Mediator analyzes the request by looking at the message signature and determining the kind of request. Since the request is a service query it then, forwards the requests to the MatchMaker registries. On receipt of the response, the Information Mediator forwards the resource query with the requested resource properties to MDS. The result of the resource query shows the set of resources which serve the specific service and properties the client is looking for. The Information Mediator selects only services served by those resources from the service query result and, finally, returns the service list to the client. The Information Mediator can narrow down the service list by combining resource information. This means that the client can get the service list that it actually wants to use.

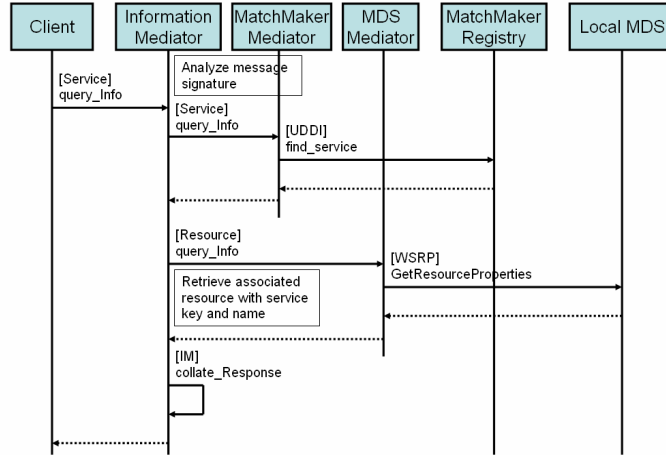


Fig. 5. Sequence Diagram – Query for a Service

In this sequence of operations, the client was unaware of the association between the service and the resource and was able to get the information request via the unified UME API.

5 Experiments

In order to evaluate the performance of our prototype, it is very important to capture the distinct characteristic of the dynamic resource. The proposed architecture doesn't improve the response time and measuring the time of query doesn't highlight the architecture. However, considering effective query hit ratio can clearly demonstrate the efficiency of this approach. The effective query hit means finding real services results which user really wants to use after submitting the service query.

We captured query hit rate of service discovery request using the proposed Information Mediator architecture, Semantic MatchMaker and MDS. We used assumption as; all the services have different identity and equally distributed three different profiles, each node is randomly hosting the chosen services and the service requestor wants to find a service profile which has CPU utilization less than 30%. Furthermore, services are randomly distributed on the nodes and some resources do not host the services.

As we can see in figure 6, Solid line shows linear yield of search result for the given service using only UDDI. Even though services have different capabilities, the UDDI query simply yields all the services cause of their identical service syntax. The square dot line shows almost linear yield of search result for all nodes which may or may not host the service. The Globus MDS query results are performed against the CPU utilization. The dashed double dot line shows reduced number of search result for the

given service using only Semantic MatchMaker. The OWL-S query is able to narrow down the results because not all services have identical capability. The dashed dot line indicates number of search results taking into account both the service and the resource capabilities. The result is remarkable because it even reduced OWL-S query result further after applying resource capabilities.

Our experiments shows that proposed architecture can effectively narrow down the query results and considerably increase the degree relevancy of service discovery results.

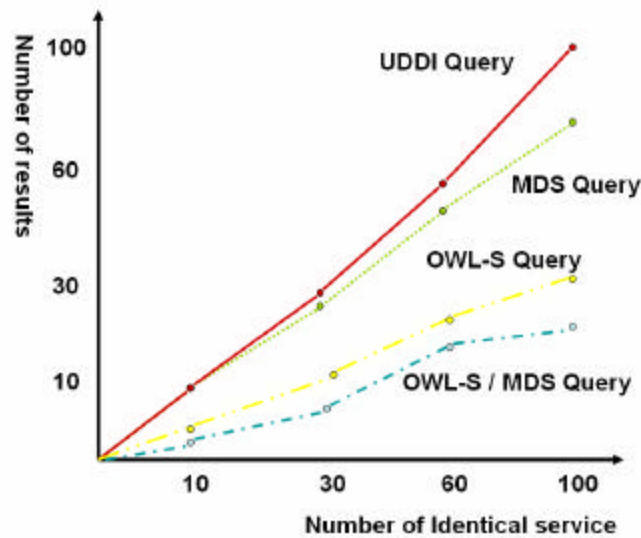


Fig. 6. Comparison of search hit ratio of proposed architecture

6 Related Work

In[6], the authors proposed DHT based Grid information service which can be understand to be lie at the intersection of P2P and Grid computing. Authors claimed that proposal goes beyond MDS because they rely on expressive semantic-based data model and P2P based decentralized information service. But the authors failed to show their semantic information model. Proposal made an assumption of existence of RDF on semantic data model however common information model is such an important component to be simply made assumption And the authors didn't explain how decentralized P2P information service can maintain dynamic state of distributed Grid resources which Globus MDS does.

In[7], the authors proposed the knowledge layer on top of Grid Resource Broker. Within the paper, authors addressed the drawbacks of MDS by pointing out its

symmetric, attribute based matching service. But their proposal is based on Gridbus broker which is proprietary middleware and furthermore, knowledge layer is integrated into Gridbus middleware. That approach will weaken compatibility and flexibility of Grid computing. Our approach for semantic service and resource discovery is totally independent to Grid middleware and any specific information service. Our proposal can be extended to support various type of Grid middleware and information service.

7 Conclusions and Future Work

In this paper, we have presented a distributed service discovery architecture that provides a single interface for the semantic discovery of the service and resource. Our architecture enables any service or resource to be discovered in a uniform manner. The service discovery clients can take advantage of the single discovery API and retrieve a more relevant result set. Implementation of the architecture demonstrates simple service discovery operation and enables seamless integration with existing discovery infrastructure.

The architecture described in this paper addressed semantic descriptions and properties of services and resources that can be used to discover them. But it still covers only the limited amount of resource definition. Adding network knowledge to service discovery API will enable more effective and robust searches thereby improving usability of the API. We intend to explore how semantics can be folded into network resource management next.

We'll also look into emerging Web Service technologies such as WS Addressing, WS Policy and UDDI property representation. Those evolving new technologies will enhance our architecture by addressing service capability and property within UDDI data structure.

References

1. Luc Moreau, Simon Miles, Juri Papay, Keith Decker and Terry Payne : Publishing Semantic Descriptions of Services
2. Ludwig, Simone A.; Reyhani S.M.S : Semantic Approach to Service Discovery in a Grid Environment
3. E. Benson, G. Wasson, and M. Humphrey: Evaluation of UDDI as a Provider of Resource Discovery Services for OGSA-based Grids. 2006 International Parallel and Distributed Processing Symposium (IPDPS 2006)
4. UDDI Version 3.0.2: http://uddi.org/pubs/uddi_v3.htm
5. Information Services (MDS): <http://www.globus.org/toolkit/docs/4.0/info/key/>
6. Manolis Koubarakis, Zoi Kaoudi, Iris Miliaraki, Matoula Magiridou and Antonios Papadakis-Pesaresi: Semantic Grid Resource Discovery using DHTs in Atlas
7. Thamarai Selvi Somasundaram, R.A.Balachandar, Vijayakumar Kandasamy, Rajkumar Buyya, Rajagopalan Raman, N.Mohanram and S.Varun : Semantic-based Grid Resource Discovery and its Integration with the Grid Service Broker

8. Frederick Lee, Garg Shishir, Garg Sukesh, Prmaila Mullan : Grid Service Discovery Integrating UDDI and MDS
9. Common Information Model (CIM) in DMTF : <http://www.dmtf.org/standards/cim/>