COMPARISON OF CENTRALIZED AND DECENTRALIZED SERVICE DISCOVERY IN A GRID ENVIRONMENT

Simone A. Ludwig
Brunel University
Uxbridge, Middlesex UB8 3PH
UK
Simone.Ludwig@brunel.ac.uk

Abstract

In this paper three different Service Discovery approaches based on centralized and distributed registries within so called Virtual Organizations are assessed. The first approach is a centralized model, the second a completely decentralized and the third is a hybrid of both models. The Grid environment is typically comprised of heterogeneous resources over wide-area networks. It addresses all distributed computing issues, especially the problem of service discovery. The benefits and drawbacks of all three models are discussed and a comparison regarding issues such as administration, management, scalability and security is followed. Furthermore, measurements are executed in order to investigate the performance of all three models, and a reliability analysis is conducted.

Key Words

Service Discovery, Service Registry, Performance Measurements, Reliability Analysis.

1. Introduction

In today's network-oriented computing environments it is common to find applications that distribute their workload over multiple hosts. In some cases, distributed applications are composed of components, each capable of residing on any arbitrary host that provides the requisite services [1]. With application components interoperating from different network hosts, efficient task distribution and communication becomes very important. In relation to distributed components communicating over the network, it is sometimes more efficient to transport an application component to a remote location, have it performed there and the result returned.

Service discovery systems enable network devices, applications and services to seek out and find other complementary network devices, applications and services which are needed to appropriately complete specified tasks. When businesses offer various types of heterogeneous services to their customers, finding a service efficiently becomes important [2]. To provide close and customized service request matches, a semantic matching process based on defined ontologies would be an option [3].

Grid computing [4] has primarily focused on solving the systems-management challenges of distributed computing, such as security, authentication and policy management across heterogeneous platforms and organizations. Utility computing has focused on developing provisioning technology and the business model for on-demand, pay-as-you-go infrastructure [5]. Grid computing has been receiving a lot of attention in recent years. A Grid environment typically comprises of heterogeneous resources over a wide-area network. These resources and services are available for Grid applications and therefore, Service Discovery (SD) is an important issue.

Presently, there is effort underway to standardize services. The Open Grid Services Architecture (OGSA) [6] was created to define a base framework of services to achieve interoperability between different Grid implementations. In a Grid environment, there is a need to integrate services across distributed, heterogeneous, dynamic Virtual Organizations (VOs) formed from the disparate resources within a single enterprise and/or from external resource sharing and service provider relationships. This integration can be technically challenging because of the need to achieve various qualities of service when running on top of different native platforms. Building on concepts and technologies from the Grid and Web services communities, this architecture defines uniform exposed service semantics, so called the Grid service. It provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities. OGSA also defines, in terms of Web Services Description Language, interfaces and associated conventions, mechanisms required for creating and composing sophisticated distributed systems, including lifetime management, change management and notification [7].

Though, OGSA is defined to provide interoperability and discovery of services between different Grid domains, SD is still considered as a centralized approach of having

one central registry within each VO. As the Grid is distributed by nature, SD for distributed resource sharing/resource allocation is not satisfactory in a centralized model. As it can be seen in the next section, the Centralized Service Discovery (CSD) approach does not scale well and therefore, there is a need to provide decentralized approaches as well.

This paper is based on the current work being done for the DataTAG [8] project. The goal of the DataTAG project is to create a large-scale intercontinental testbed for data-intensive Grids and to provide interoperability between the Grid middleware layers.

The remainder of this paper is organized as follows. Section 2 gives an overview of three different SD approaches including detailed advantages and disadvantages of each. Section 3 is concerned with performance measurements and the evaluation of the results. In section 4, a reliability analysis is conducted. Section 5 contains the results of the comparison and section 6 concludes this paper.

2. Three Different Approaches for Service Discovery

While many service discovery systems exist, most have not been designed to operate in a pervasive manner so that they can span the many complex and different computing environments that exist in today's computing infrastructure. Some service discovery systems are limited to certain network or computing environments, many are too tightly defined for a specific usage, and others do not provide the necessary flexibility in their discovery approach.

In a Grid environment, resources and services are owned by various administrative organizations and shared under locally defined policies that specify what is shared, who is allowed to share and under what conditions [9]. A set of individuals and/or institutions defined by such sharing rules is defined as a VO. Resource discovery and therefore also SD in a Grid is made challenging by the potentially large number of resources and users, and the heterogeneity in resource types and user requests. Resource discovery is further complicated by the natural tendency for VOs to evolve over time, with e.g. institutions joining and leaving, along with their resources and users. Furthermore, the number of resources shared by an institution varies and also the resources' characteristics such as availability and CPU load.

2.1 Centralized Service Discovery

CSD contains only one global registry where every service of each VO registers itself. This is shown in Figure 1. (In the following four figures only 3 VOs are illustrated to simplify the structure.)

The advantage of having one global registry is that the management of the distributed resources is done centrally. The control of authority and quality is easily managed and the service description is consistence in style and presentation. The efficient usage of resources is organized via the global registry. Authorization and security issues are handled quite easily as they are administered from a central point.

The disadvantages are as follows. This approach does not scale very well as there is a limit for the number of resources which are managed and registered in the global registry. Furthermore, parallel accesses to the registry are very limited and the response times could be very high for a high workload.

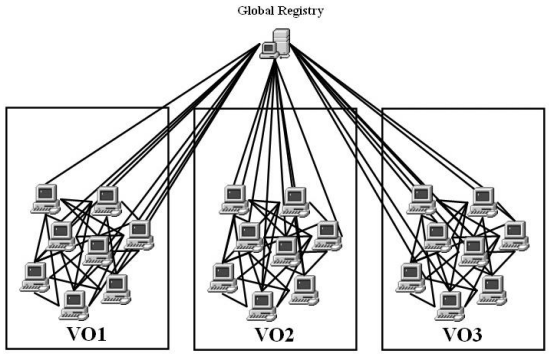

Figure 1: CSD Approach

This approach may be good for a very limited number of resources but not for a large-scale environment such as the Grid. In order to guarantee a fully functional system, the global registry must be replicated to prevent the case of failure.

2.2 Decentralized Service Discovery

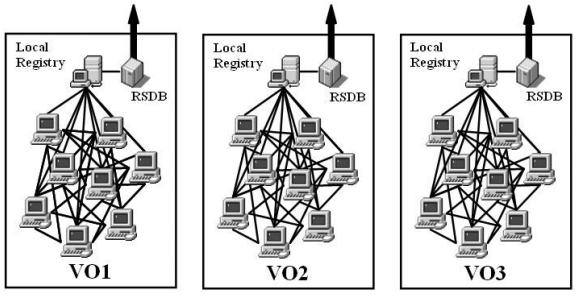

Figure 2: DSD Approach with RSDB

In the Decentralized Service Discovery (DSD) model, there are only local registries, one for each VO. There are two different models; one is shown in Figure 2 consisting of a local registry and a Remote Server Database (RSDB) for each VO. This RSDB holds a list of all remote VO

servers it is authorized to. The second model, also called chain model (see Figure 3), contains a local registry for each VO and is directly connected to the next VO in line.

The advantage of these two approaches is that there is no management of a central registry required anymore. That implies that this part of the administration is not necessary as the administration is done locally for every VO. Issues regarding security, complexity and authorization are managed for each VO independently.

A problem of these approaches arises when a service request is not satisfied within a VO. In this case, the own VO has to lookup the service in another VO. Here, the two models have to be differentiated. In the first model, a list of various VO servers is stored in the RSDB. When a service is not found locally, e.g. in VO1, the local registry looks into the RSDB and contacts the next VO server in the list (e.g. VO2). If the service is also not found on this machine, the request is sent back to VO1 and the next entry in the RSDB list is chosen. If this list finishes and the service is not found, the service requester contacts the RSDB list and looks for the next VO, which in this case is VO2. The service requester then looks for another server where no request was sent to so far and so on. If the requested service is not found, there has to be some kind of timeout agreement in order to prevent the user having to wait endlessly. Another disadvantage is the RSDB. This needs to be updated and managed within a VO which might lead to unnecessary replication. In the event of a failure in the local registry within a VO, the next VO in the RSDB list is contacted.

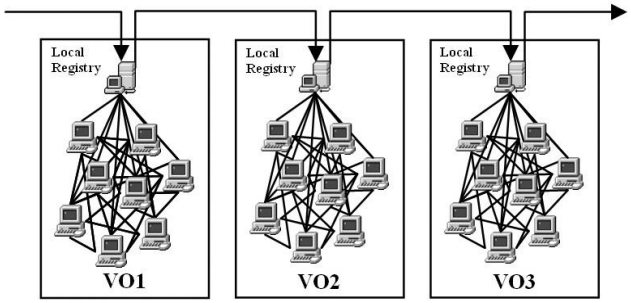

Figure 3: DSD Chain Approach

For the chain model, the service request is sent by sending it from one “chain” (VO) to the other. If a service is not found locally or on the closest VO server, the request is send along the chain until the appropriate service provider is found. The event of a failure in the local registry within a VO is disastrous for this model. The only way to proceed in this case is to go along the chain in the opposite direction assuming that this path does not encounter any problems.

2.3 Hybrid Service Discovery

This approach is a hybrid of the centralized and decentralized model as shown in Figure 4. Each VO has its own local registry and the whole organization is managed by one global centralized registry.

This mixed approach has the benefit of the shared management and administration of local and global registries. The VOs are responsible for the management of their own resources but the global registry is responsible for high-level management. When a service is absent in the local registry within the VO, a request is sent to the global registry inquiring where to find this particular service. It then can directly contact the responsible VO. Issues such as authorization, security and complexity are easily handled. With respect to scalability, this approach scales well up to the limit of storage capacity of the global registry. But having several sets of global and local registries reduces the limitation of the global registry.

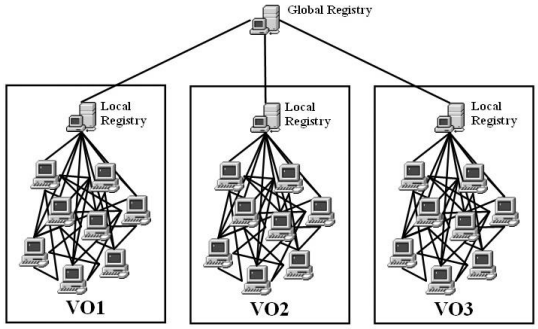

Figure 4: HSD Approach with Centralized Registry

The Hybrid Service Discovery (HSD) approach has the drawback that the resources are administered twice, in the local registries of the individual VOs and also in the global registry. If the global registry fails, there is no communication between VOs possible anymore. Therefore, a replicated global registry is necessary to guarantee a fully functional system.

3. Performance Measurements

In order to conduct performance measurements for the three different SD approaches, a prototype was developed. This prototype is based on Web Services (WSs) technology standards in alignment to the OGSA standard. The implementation of the WSs was performed in Java using WSDL (Web Service Description Language), XML (Extensible Markup Language) and SOAP (Simple Object Access Protocol). SOAP and WSDL are designed to provide descriptions of message transport mechanisms in order to describe the interface used by each service [10]. A service registry, UDDI (Universal Description, Discovery and Integration) was used. UDDI is another emerging XML-based standard to provide a registry of businesses by their physical attributes such as name, address and the services that they provide [11]. In addition, UDDI descriptions are

augmented by a set of attributes that are called TModels, which describe additional features such as the classification of services within taxonomies.

The prototype works as follows. The service requester sends out a search request to a UDDI registry and the result is returned. The measurement of the Service Discovery Time (SDT) is basically done by taking the difference of two time stamps when the service request is made.

Figure 5 shows the measurement setup for the three approaches. In registry 1, WS1, WS2 and WS3 were registered. In registry 2, WS4 and WS5, in registry 3, WS6 and WS7 and in registry 4, WS8 and WS9 were stored. One exception is that for the centralized approach all WSs were stored in registry 1. All measurements were executed 100 times in a set of 10 in order to achieve good average result values.

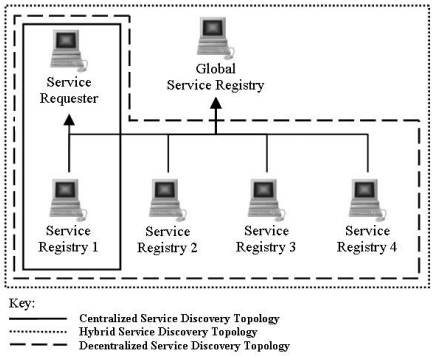

Figure 5: Measurement Setup

To see if the measurements reflect the theoretical analysis, the following equations were derived.

For the CSD approach, the SDT is defined by

$$t_{CSD} = T_{request} + T_{search} + T_{response}. \quad (1)$$

The SDT for the DSD approach is calculated by

$$t_{DSD} \leq T_{request} + T_{search} + nT_{conN\,Reg} + nT_{search} + T_{response}. \quad (2)$$

For the HSD approach, the SDT is given by

$$t_{HSD} \leq T_{request} + T_{search} + T_{conG\,Reg} + T_{search} + T_{response}. \quad (3)$$

Whereas:

n: Factor indicating number of registry queries, when service is not found in registry where the request was sent to ($n \geq 0$).
$T_{request}$: Request time.
T_{search}: Search time in registry.
$T_{response}$: Response time.
$T_{conNReg}$: Time to contact the next registry in line.
$T_{conGReg}$: Time to contact the global registry.

From equations (1) – (3), the SDTs derive the following expressions

$$t_{DSD} \geq t_{CSD} \text{ and } t_{DSD} \geq t_{HSD}. \quad (4)$$

The following measurements are achieved. Figure 6 shows the SDTs for the CSD approach. All WSs were registered globally in registry 1. The average outcome of the SDT is 140ms with a variation of ± 2ms.

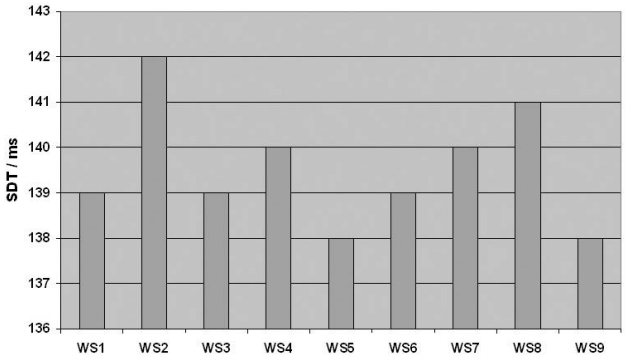

Figure 6: Performance Measurements for CSD

Figure 7 gives an account of the measurements of the DSD approach. The chain model was chosen and the graph shows increasing distribution. Requests were always made to registry 1 first, where WS1, WS2 and WS3 reside, in order to measure the maximum value when looking for WS8 and WS9. Finding these services (WS1, WS2 and WS3) give the shortest SDTs, whereas WS4 and WS5 (residing in registry 2) give greater values. The same pattern occurs for the pair WS6/WS7 and WS8/WS9 as all registries in the chain are contacted if the service is not found earlier on in a registry. The maximum SDT is found for WS8 and WS9 as those services reside in the last registry of the chain.

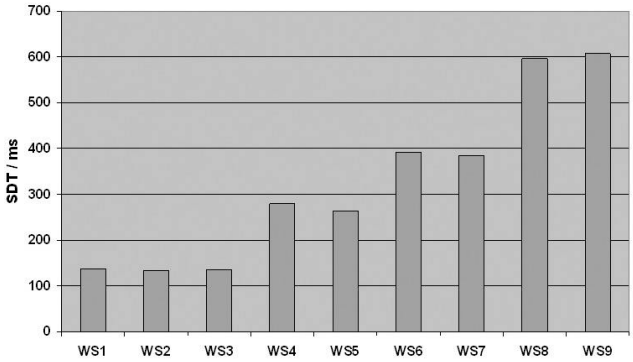

Figure 7: Performance Measurements for DSD

Figure 8 shows the measurements of the HSD approach. The shorter columns show the SDTs when the services are found in the registry where the request was sent to. The longer columns show the SDTs when the service was not found in the registry the request was sent to, but in the global registry respectively. It can be seen that the measurements taken from registry 4 are greater than those performed in registry 1, 2 or 3. This can be also seen in Figure 7, where the increase for WS8 and WS9 is comparatively greater than the previous ones. This is due to the fact that the first three machines (registry 1-3) and

the last machine (registry 4) were situated at two different locations. The reasons for the constant increase are network connectivity, hardware and software configuration and the instantaneous load distribution on each machine. The hardware and software configuration can be thought of being negligibly small. The work load on the machines can also not be so significant as that would mean that the work load always coincides with the measurements. Therefore, the main reason for the constant increase is the network connectivity at the two different locations.

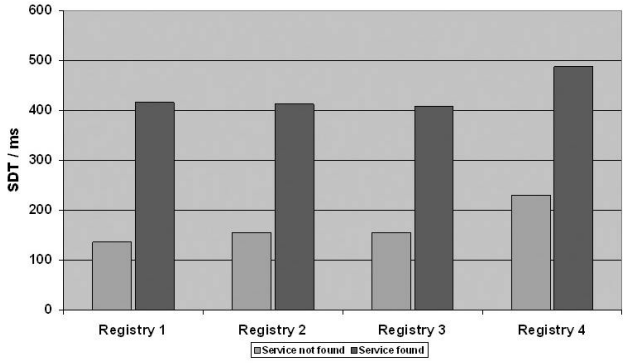

Figure 8: Performance Measurements for HSD

A comparison of the three different SD approaches was accomplished and the minimum and maximum values of the SDTs are shown in Figure 9. It can be stated that the predicted inequalities in (4) are valid as the measurements show the same results qualitatively. The minimum SDT of all three approaches is nearly the same as for every model the service request is sent to at least one registry. The maximum SDT for DSD is equal or greater than for HSD and greater than for CSD depending on the number of registries in line. For HSD, the maximum SDT is also greater than for CSD.

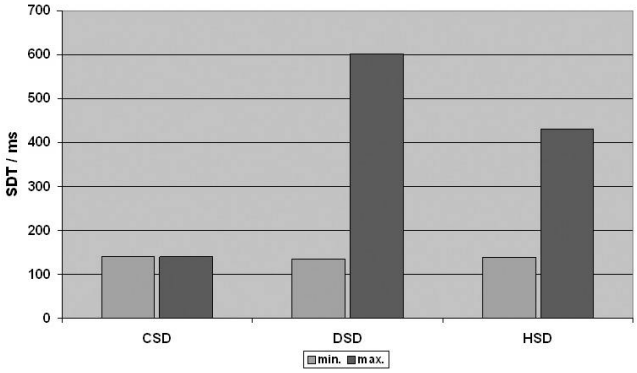

Figure 9: Comparison of Minimum and Maximum Values

4. Reliability Analysis

The analysis of the reliability of the three different SD approaches is stated below. The following equations give an account of how reliable those systems are in case of a registry failure.

The probability of the CSD is the probability that the service is found in the registry multiplied by the factor that the registry is working. This is defined by

$$P_{CSD} = p_{CR} \cdot (1 - r_{CR}) . \quad (5)$$

For the DSD, the probability results in the sum of the probability of each decentralized registry multiplied by the product of the probabilities that the registries are working. This is given by

$$P_{DSD} = \sum_{i=1}^{n} p_i \cdot \prod_{j=1}^{i} (1 - r_j) . \quad (6)$$

Assumption: Registries are independent i.e. service might or might not exist.

The probability of the HSD is defined by

$$P_{HSD} = p_{DR} \cdot (1 - r_{DR}) + (p_{CR} - p_{DR})(1 - r_{DR})(1 - r_{CR}) . \quad (7)$$

The first part of the equation gives the probability that the service is found in the decentralized registry. The second part takes into account the probability that the service exists in the central registry but not in the decentralized registry, multiplied by the probability that the decentralized registry is working and the probability that the centralized registry is also working.

Whereby:

p_{CR}: Probability that service is found in the central registry.

p_{DR}: Probability that service is found in the decentralized registry where the request was sent to.

p_i: Probability that service is found in registry i.

r_{CR}: Probability that centralized registry is not working.

r_{DR}: Probability that decentralized registry is not working.

r_j: Probability that registry j is not working.

Analyzing all three SD approaches shows the CSD having the lowest reliability. In case of a service registry failure the whole system breaks down. The comparison of the DSD with the HSD approach results in the hybrid version having a greater reliability than the DSD, because of the replication of the services in the global registry.

5. Comparison of Results

Grid environments cannot rely on having a single device permanently present, even if it is replicated. Thus, this centralized approach is not feasible for a large environment. Furthermore, it does not scale well if more VOs are entering the community. The decentralized and the hybrid approach seem to be more practical since the reliability is greater than the one for the centralized approach and the scalability is warranted. Table 1 shows the comparison criteria. It can be seen that the reliability is best in the hybrid version. Unfortunately, the number of entries in the global registry is limited in this model, but

on the other hand, the number of accesses possible to the registries is far greater than for the centralized model.

	CSD	*DSD*	*HSD*
Administration	Easy	More difficult	More difficult
Management	Easy	More complex	More complex
Security	Easy	More complex	More complex
Scalability	Not good	Good	Good
Performance	$t_{CSD} \leq t_{DSD}$	$t_{DSD} \geq t_{HSD}$	$t_{HSD} \geq t_{CSD}$
Reliability	Lowest	Medium	Highest

Table 1: Characteristics of the Service Discovery Approaches

6. Conclusion

There are always benefits and drawbacks to take into consideration when choosing an appropriate SD model for a particular distributed Grid environment depending on requirements and restrictions.

Due to scalability reasons, the centralized approach is not feasible for a Grid environment and would only serve the need of a single VO. The decentralized and hybrid models are better not only as the scalability is warranted, but they also have a better error recovery. Error recovery is an essential issue for a fully functional SD system and is expressed by reliability, which is best in the hybrid approach.

7. Acknowledgement

I would like to thank everyone supporting this work and collaborating in the DataTAG project. This research is funded by PPARC (Particle Physics and Astronomy Research Council, UK) through the IST program of the European Union (Grant IST-2001-32459) and forms part of work package 4 of the DataTAG project.

The author would like to thank Dr. M. Reyhani for his helpful suggestions. Furthermore, I would like to thank Prof. P. Clarke for his help. Special thanks to the DataTAG members Mr. P. Moroni, Mr. E. Martelli, Dr. R. Tasker and Mr. G. Rorato for providing me with machines for the performance measurements.

References

[1] J. Smith, *Distributed computing with Aglets*, White Paper. http://www.vistabonita.com/papers/DCAglets/Introduction.html.

[2] S.A. Ludwig, *Review of various Service Discovery Systems*, Technical Report, TR-DGRG695, Department of Electronic and Computer Engineering, Brunel University, UK (2002).

[3] S.A. Ludwig, An Ontology-based Service Discovery Matchmaking Framework, *accepted for publication in Special Issue on Semantic Grid and Knowledge Grid, Future Generation Computer Systems (FGCS 2003)*.

[4] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann, 1999).

[5] O. Malik, *Distributed computing grid networks*, White Paper. http://www.redherring.com/insider/2002/10/network-computing-101102.html.

[6] I. Foster et al., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Global Grid Forum, June 2002. http://www.globus.org/research/papers/ogsa.pdf.

[7] D. Gannon, K. Chiu, M. Govindaraju and A. Slominski, *A Revised Analysis of the Open Grid Services Infrastructure*. http://www.extreme.indiana.edu/~gannon/ogsaAnalysis4.pdf.

[8] DataTAG Project. http://www.datatag.org.

[9] A. Iamnitchi and I. Foster, On Fully Decentralized Resource Discovery in Grid Environments, *International Workshop on Grid Computing*, November 2001, Denver, CO.

[10] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura and R. Neyama, *Building Web Services with Java* (SAMS, 2001).

[11] *UDDI*, Technical White Paper, September, 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.