Towards a Meta-Negotiation Architecture for SLA-Aware Grid Services

Ivona Brandic
Institute of Information Systems
Vienna University of Technology
Argentinierstraße 8
1040 Vienna, Austria
ivona@infosys.tuwien.ac.at

Srikumar Venugopal, Michael Mattess, and Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{srikumar,mmattess,raj}@csse.unimelb.edu.au

Abstract

In novel market-oriented resource sharing models resource consumers pay for the resource usage and expect that non-functional requirements for the application execution, termed as Quality of Service (QoS), are satisfied. QoS is negotiated between two parties following the specific negotiation protocols and is recorded using Service Level Agreements (SLAs) standard. However, most of the existing work assumes that the communication partners know about the SLA negotiation protocols and about the SLA templates before entering the negotiation. However, this is a contradictory assumption, if we consider computational Grids and novel commercially oriented Computing Clouds where consumers and providers meet each other dynamically and on demand. In this paper we present novel meta-negotiation and SLA-mapping solutions for Grid workflows bridging the gap between current QoS models and Grid workflows, one of the most successful Grid programming paradigms. We illustrate the open research issues with a real world case study. Thereafter, we present document models for the specification of meta-negotiations and SLA-mappings. We discuss the architecture for the management of meta-negotiations and SLA-mappings as well as integration of the architecture into a Grid workflow management framework.

General Terms Grid Computing, Distributed Computing, Service Level Agreements (SLAs)

Keywords Quality of Service (QoS), Grid Economy, SLA-based Grid services, Meta-negotiation

1. Introduction

Grid computing enables geographically distributed and heterogeneous computational and storage resources to be accessed across administrative domains in order to efficiently solve large scale scientific problems. Present-day Grids are based on quid pro quo arrangements wherein partners within scientific collaborations provide access to each other's resources. In such Grids, participants usually do not have guarantees for obtaining resources whenever they want and do not pay for resource usage either. However, with the maturity of Grids, users have begun to expect that more sophisticated requirements, specified by means of Quality of Service (QoS) parameters, are satisfied. Typically, such requirements relate to user experience, such as completion of job execution within a specific deadline, or a guarantee on bandwidth for data transfer. Users establish Service Level Agreements (SLAs) with resource providers which guarantee that QoS requirements will be met in exchange for appropriate remuneration.

Before committing themselves to an SLA, the user and the provider may enter into negotiations that determine the definition and measurement of user QoS

parameters, and the rewards and penalties for meeting and violating them respectively (3; 6). The term *negotiation strategy* represents the logic used by a partner to decide which provider or consumer satisfies his needs best. A *negotiation protocol* represents the exchange of messages during the negotiation process. Recently, many researchers have proposed different protocols and strategies for SLA negotiation in Grids (20; 3; 5; 11). However, these not only assume that the parties to the negotiation understand a common protocol but also assume that they share a common perception about the goods or services under negotiation. In reality however, a participant may prefer to negotiate using certain protocols for whom it has developed better strategies, over others. Also, a participant may choose to only allow certain aspects of a good or a service to be negotiated which may not be acceptable to others. In other words, the parties to a negotiation may not share the same understanding that is assumed by the earlier publications in this space.

In order to bridge the gap between different negotiation protocols and scenarios, in this paper, we propose a so-called *meta-negotiation* architecture. Meta-negotiation is defined by means of a *meta-negotiation document* where participating parties may express: the pre-requisites to be satisfied for a negotiation, for example a specific authentication method required or terms they want to negotiate on (e.g. time, price, reliability); the negotiation protocols and document languages for the specification of SLAs, e.g. Web Service Level Agreement (WSLA) (22) or WS-Agreement (23) that they support; and conditions for the establishment of an agreement, for example, a required third-party arbitrator. These documents are published into a searchable registry through which participants can discover suitable partners for conducting negotiations. In our approach, the participating parties publish only the protocols and terms while keeping negotiation strategies hidden from potential partners. Meta-negotiation approach allows two parties to reach an agreement on what negotiation protocols and documents to use before starting the negotiation process.

The main contributions of this paper are therefore: (i) development of the architecture for the *meta-negotiations* in Grid systems; (ii) description of the *meta-negotiation document*; and (iii) demonstration of the usability of the meta-negotiation framework for real-world Grid negotiations.

The rest of this paper is organized as follows: Section 2 presents the related work on Grid and Web service negotiations. Section 3 describes phases of the proposed meta-negotiation as well as the meta-negotiation document in detail. Section 4 discusses the meta-negotiation architecture including registry, meta-negotiation middleware, service provider, and service consumer. In Section 5 we present the evaluation of the meta-negotiation approach. Section 6 presents our conclusions and describes the future work.

2. Related Work

Currently large body of work has been done in the area of Grid service negotiation and Quality of Service.

Quan et al., Ouelhadj et al., and Elmroth et al. discusses incorporation of SLA-based resource brokering into existing Grid systems (15; 14; 6). Wieczorek et al. proposes a novel approach for modeling scheduling problems as an extension of the multiple-choice knapsack problem (17). A general bi-criteria scheduling heuristic is proposed called Dynamic Constraint Algorithm (DCA) based on dynamic programming. Li et al. discusses Rudder framework, which facilitates automatic Grid service composition based on semantic service discovery and space based computing (12).

Venugopal et al. propose a negotiation mechanism for advance resource reservation using the alternate offers protocol (20). However, it is assumed that both partners understand the alternate offers protocol. Brandic et al. proposes a holistic Grid infrastructure for specification, planing and execution of QoS aware Grid workflows (3). Services are selected based on integer programming approach, whereas service negotiation is performed using WSLA and implicit negotiation protocol. Similar to (20) in (3) is assumed that each participating service understands the necessary negotiation protocol.

Work presented in Al-Ali et al. extends the service abstraction in the Open Grid Services Architecture (OGSA) for QoS properties focusing on the application layer (1). Thereby, a given service may indicate the QoS properties it can offer or it may search for other services based on specified QoS properties. Work presented in Czajkowski et al. proposes generalized resource management model where resource interactions are mapped onto a well defined set of platform-independent SLAs (5). The model is based on Service Negotiation and Acquisition Protocol (SNAP) provid-

ing the lifetime management SLAs. SNAP is embedded into the Globus Toolkit.

Hill et al. discusses an architecture that allows changes to the Grid configuration to be automated in response to operator input or sensors placed throughout the Grid based on principles of autonomic computing (9). Similarly to Hill et al. work discussed in Vambenepe et al. addresses global service management based on principles of autonomic computing (18).

Vu et al. present an extensible and customizable framework for the autonomous discovery of semantic Web services based on their QoS properties (21). FIPA Abstract Architecture Specification proposes an abstract architecture for the negotiations based on agent systems (7). However, FIPA does not address implementation issues of negotiation systems. Condor's ClassAds mechanism is used to represent jobs, resources, submitters and other Condor daemons (16).

Work presented in Iyer et al. shows how scalability of grids can be enhanced by adopting peer-to-peer (P2P) techniques in order to implement decentralized grid services (10). Narendra discusses the issues of generating contracts used to govern inter-service interactions (13). In particular he investigates whether the established contracts are beneficial and/or safe from a participants perspective. Xiong et al. demonstrates how individual services can be federated into composite services which are able to execute a given task subject to service level agreements (SLA) (24).

To the best of our knowledge none of the presented approaches address *meta-negotiations (MN)* where participating parties may agree on a specific negotiation protocol, security standards or other negotiation pre-requisites. In our approach we address meta-negotiations where participating parties may specify negotiation requirements and based on a private selection strategy select those services which promise successful negotiation.

3. Meta-negotiation Framework

In this section, we present an example scenario for the meta-negotiation architecture and describe the document structure for publishing negotiation details into the meta-negotiation registry.

3.1 Scenario

As depicted in Figure 1, the meta-negotiation infrastructure can be employed in the following manner:

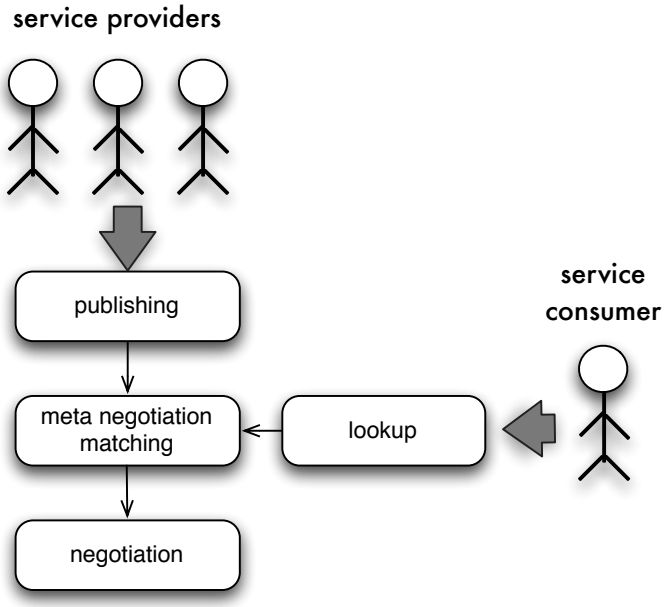

Figure 1. Meta-negotiation phases

Publish. A service provider publishes descriptions and conditions of supported negotiation protocols into the registry (see Section 4).

Lookup. Service consumers perform lookup on the registry database by submitting their own documents describing the negotiations that they are looking for.

Match. The registry discovers service providers who support the negotiation processes that a consumer is interested in and returns the documents published by the service providers.

Negotiate. Finally, after an appropriate service provider and a negotiation protocol is selected by a consumer using his/her private selection strategy, negotiations between them may start according to the conditions specified in the provider's document.

Note that in this scenario, the consumer is looking for an appropriate service provider. The reverse may happen as well, wherein a consumer advertises a job or a task to be carried out and many providers bid to complete it. In such cases, the providers would perform the lookup.

3.2 Registry Document

The participants publishing into the registry follow a common document structure that makes it easy to discover matching documents. This document structure

```
1.  <meta-negotiation
2.     xmlns:xsi="..."
3.     xsi:noNamespaceSchemaLocation="...">
4.     <entity>
5.       <contact name="..."
6.         phoneNumber="..." />
7.       <organization
8.         name="University of Melbourne"
9.         ...
10.     <ID name="1234"/>
11.   </entity>
12.   <pre-requisite>
13.     <role name="consumer"/>
14.     <security>
15.       <authentication value="GSI
16.         location="uri"/>
17.     </security>
18.     <negotiation-terms>
19.       <negotiation-term name="beginTime"/>
20.       <negotiation-term name="endTime"/>
21.       <negotiation-term name="price"/>
22.     </negotiation-terms>
23.   </pre-requisite>
24.   <negotiation>
25.     <document name="WSLA" value="uri"
26.       version="1.0" />
27.     <document name="WS-Agreements"
28.       value="uri" version="1.0" />
29.     <protocol name="alternateOffers"
30.       schema="uri" version="1.0"
31.       location="uri"/>
32.   </negotiation>
33.   <agreement>
34.     <confirmation name="arbitrationService"
35.       value="uri"/>
36.   </agreement>
37. </meta-negotiation>
```

Figure 2. Example document for meta-negotiation registry

is presented in Figure 2 and consists of the following main sections. Each document is enclosed within the `<meta-negotiation> ... </meta-negotiation>` tags. The document contains an `<entity>` elements defining contact information, organization and ID of the participant. The `<ID>` element defines the unique identifier given to the meta-negotiation document by the registry. The publisher can update or delete the document using the identifier. Each meta-negotiation comprises three distinguishing parts, namely *pre-requisites*, *negotiation* and *agreement* as described in the following paragraphs.

Pre-requisites. The conditions to be satisfied before a negotiation are defined within the `<pre-requisite>` element (see Figure 2, lines 12–23). Pre-requisites define the *role* a participating party takes in a negotiation, the *security credentials* and the *negotiation terms*. The `<role>` element defines whether the specific party wants to engage in the negotiation as a provider or as a consumer of resources. The `<security>` element specifies the authentication and authorization mechanisms that the party wants to apply before starting the negotiation process. For example, in Figure 2, the consumer requires that the other party should be authenticated through the *Grid Security Infrastructure (GSI)* (8) (lines 15–16). The negotiation terms specify QoS attributes that a party is willing to negotiate and are specified in the `<negotiation-term>` element. For example, in Figure 2, the negotiation terms of the consumer are *beginTime*, *endTime*, and *price* (lines 19–21).

Negotiation. Details about the negotiation process are defined within the `<negotiation>` element. In Figure 2, the consumer supports two document languages and one negotiation protocol. Each document language is specified within `<document>` element. In Figure 2, *WSLA* and *WS-Agreements* are specified as supported document languages. Additional attributes specify the *URI* (Uniform Resource Indicator) to the API or WSDL for the documents and their versions supported by the consumer (lines 25–26). In Figure 2, *AlternateOffers* is specified as the supported negotiation protocol. In addition to the *name*, *version*, and *schema* attributes, the URI to the WSDL or API of the negotiation protocols is specified by the *location* attribute (lines 29–31).

Agreement. Once the negotiation has concluded and if both parties agree to the terms, then they have to sign an agreement. This agreement may be verified by a third party organization or may be lodged with another institution who will also arbitrate in case of a dispute. These modalities are specified within the `<agreement>` clause of the meta-negotiation document. For example, in Figure 2, a third party service, called “arbitrationService”, is specified for confirming the agreement between the two parties.

4. A Case Study of

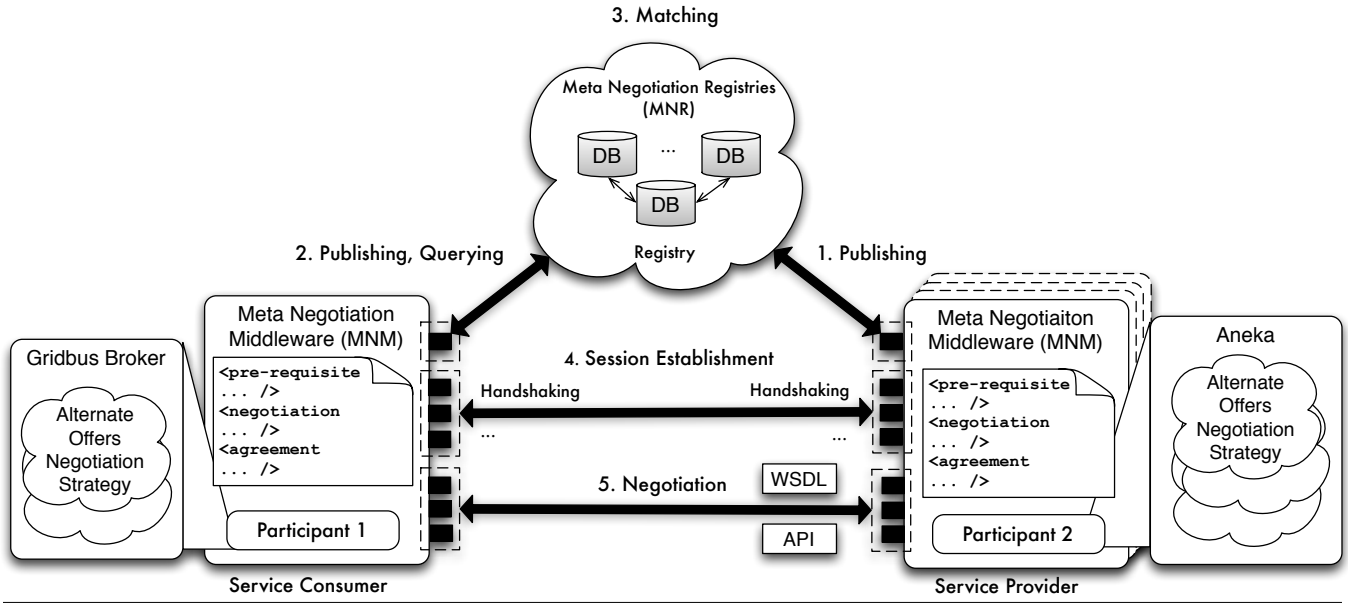

Figure 3. Architecture for meta-negotiation in heterogeneous Grids with sample provider and consumer

Meta-negotiation

In order to create a case study that tests the proposed meta-negotiation framework in practice, we have extended a previous publication on negotiation of advance reservations using the alternate offers protocol to incorporate the meta-negotiation framework (20). The architecture followed in this case study is shown in Figure 3. It consists of the registry for meta-negotiation documents and the meta-negotiation middleware on both the provider and consumer sides.

In our architecture, the service provider role is carried out by Aneka (4), which is a resource management system for enterprise Grids composed of machines running Microsoft Windows operating system. Aneka provides facilities for advance reservation of computing nodes and supports flexible scheduling of applications constructed using different parallel programming models such as bag-of-tasks and dataflow computing. The Gridbus Broker (19) maps jobs to appropriate resources considering various restrictions specified by terms of *functional* and *non-functional* requirements. *Functional requirements* include but are not limited to task and data dependencies such as, for example, a sequence of tasks is required to execute a specific application. *Non-functional requirements* include QoS parameters such as budget restrictions, and a deadline for execution. The broker can guarantee the end-user's deadline requirement only if it is able to reserve nodes on resources in advance. Therefore, in this respect, the broker functions as a consumer that requests reservations from the provider.

In our current prototype we assume that the provider and the consumer have the same semantic meaning for the terms and protocols used in context of meta-negotiation. For example, the term *beginTime* means earliest possible begin time for the execution of an application for all participants of the meta-negotiation. Semantic matching based on ontologies would be required when these terms are interpreted differently.

4.1 Registry

The registry is a searchable repository for meta-negotiation documents that are created by the participants. Currently, this is implemented as a PostgreSQL database with a web service front end that provides the interface shown in Figure 4. However, it is possible to host the registry using a cloud of databases hosted on a service provider such as Google App Engine[1] or Amazon S3. When a meta-negotiation document is published, the registry assigns it a unique identifier (ID) that can then be used for subsequent operations. The query call tries to find the documents that match the maximum number of attributes of the search query. It returns an array of IDs of these documents to the caller which can then fetch each one through the `getDocument` call.

[1] http://code.google.com/appengine

```
1. publish(XMLdocument);
2. update(XMLdocument);
3. query(XMLdocument);
4. getDocument(ID);
```

Figure 4. Registry Methods

4.2 Meta-Negotiation Middleware

The *meta-negotiation middleware* facilitates the publishing of the meta-negotiation documents into the registry and the integration of the meta-negotiation framework into the existing client and/or service infrastructure, including, for example, negotiation or security clients. Besides being as a client for publishing and querying meta-negotiation documents (steps 1 and 2 in Figure 3), the middleware delivers necessary information for the existing negotiation clients, i.e. information for the establishment of the negotiation sessions (step 4, Figure 3) and information necessary to start a negotiation (step 5 in Figure 3). As shown in Figure 3 each service consumer may negotiate with multiple service providers concurrently. As mentioned in Section 3 even the reverse may happen as well, wherein a consumer advertises a job. In such cases, the providers would negotiate with multiple consumers.

After querying the registry and applying a client-based strategy for the selection of the appropriate service, the information from the service's meta-negotiation document is parsed. Thereafter, meta-negotiation information is incorporated into the existing client software using a dependency injection framework such as Spring[2]. This dependency injection follows an Inversion of Control approach wherein the software is configured at runtime to invoke services that are discovered dynamically rather than known and referenced beforehand. This is suitable in the context of meta-negotiation wherein a participant discovers others at runtime through the registry and has to dynamically adapt based on the interfaces provided by his counterpart (usually through a WSDL document).

Figure 5 shows an example of how this would work in practice. On the consumer side, the middleware queries the registry and obtains matching meta-negotiation documents. The middleware parses the meta-negotiation document of the selected provider and dynamically injects the interfaces discovered from the WSDLs in the document for security, negotiation and arbitration services into the existing abstract clients. Currently, we support semi-automatic integration of existing clients into meta-negotiation middleware wherein the existing clients are extended with the XML-based configuration files which are then automatically populated with the discovered interfaces.

[2] http://www.springframework.org/

```
1. ...
2. <object id="WSDLParser"
3.  type="MNM.WSDLParser, MNM">
4.   <constructor-arg index="0"
5.     value="\pathToQueryingFile.xml"/>
6.   <property name="RegistryRequesterProperty"
7.     ref="RegistryRequester"/>
8. </object>
9. ...
```

Figure 6. Dependency Injection

Figure 6 shows how dependency injection can be expressed by specifying dependency injection to object *WSDLParser* (see line 2). Thereby, a meta-negotiation XML file used to query the registry is specified as a constructor argument. The location of the file can be changed without changing the code by specifying the *constructor-args* as shown in line 5 of Figure 6. WSDL location of the negotiation, security, and arbitration client can be injected in the same way.

5. Evaluation

We have evaluated the architecture presented in the previous section using actual services deployed on a real testbed shown in Figure 7. As mentioned previously, we have used the Gridbus broker as an example of a service consumer and a enterprise Grid constructed using Aneka as a service provider. The aim of this evaluation was to test the overhead of the meta-negotiation infrastructure on the overall negotiation process.

5.1 Testbed

As shown in Figure 7, we deployed the registry in a machine running Windows Server 2003. The registry was accessible through a Web service interface and used a PostgreSQL database on its backend. In a previous work (20), we evaluated a Negotiation Service for advance reservation of nodes in an Aneka Grid. Since the aim of these experiments was only to test the meta-negotiation framework, we isolated the Negotiation Service from the resource management system. Hence, it would reject any proposal for node reservation as it would not be able to determine node avail-

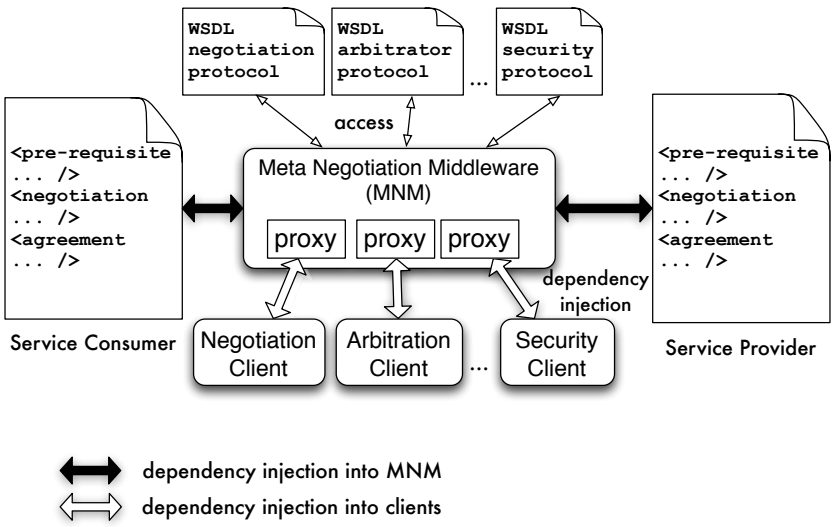

Figure 5. Meta-negotiation middleware

ability. We deployed 20 such services – *(S1,... S10)* on machines in a student lab in the Department of Computer Science and Software Engineering, University of Melbourne, Australia and *(S11,... S20)* on machines in the Department of Communication Computer and System Sciences, University of Genova, Italy. Negotiations with services located in Melbourne would terminate in single rounds (a proposal followed by a rejection). Services located in Italy would terminate after 2 retries. We published a meta-negotiation document for each service into the registry with different negotiation terms and document languages. The Gridbus broker was started on a machine in the Department of Computer Science, University of Melbourne and queried the registry in order to select an appropriate service provider. It would then open a negotiation process with the selected Aneka Negotiation Service.

5.2 Experimental Results

The results of our evaluation are shown in Table 1. As shown in Table 1 the time necessary to query the registry represents 2.91 seconds or 16.16% of the overall negotiation time. Query time is calculated as the time necessary to get the list of the IDs, i.e. invocation of the method *query(XMLdocument)*, plus the time necessary to fetch each document, i.e. multiple invocations of the method *getDocument(ID)*. The time necessary to fetch each document represents about 0.2 sec. Thus, in our experiments we fetched about 15 XML documents in average, since $2.91/0.2 \approx 15$. Please note, that all times used in Table 1 are average times measured over

	Overall Negotiation			Total
	Meta-Negotiation		Negotiation	
	Querying	Parsing		
Time in sec	**2.91**	**0.02**	**15.10**	**18.03**
Time [%]	**16.16**	**0.01**	**83.73**	**100.00**

Table 1. Experimental results

10 rounds. Time necessary to parse the selected meta-negotiation document and to inject the WSDL information into the client is 0.02 seconds or 0.01% of the overall negotiation time. Thus, time for the completion of the meta-negotiation is 2.93 seconds or 16.17% of the overall negotiation time. The time for the meta-negotiation is calculated as the the sum of the time necessary to query the registry (2.91 seconds) and the time necessary to parse the selected meta document (0.01 seconds).

The time necessary to negotiate with an *Aneka* service represents 15.10 seconds or 83.73% of the overall negotiation time. We observed that the negotiation time with services located in Italy represents about 15 seconds (due to 2 retries), since the time necessary to negotiate with services located in Melbourne represents about 5 seconds. Thus, in our experiments we have obviously negotiated only with services located in Italy. We started an alternate offers negotiation with only one round. Thus, the overall negotiation time is 18.03 seconds. Overall negotiation time is calculated as the sum

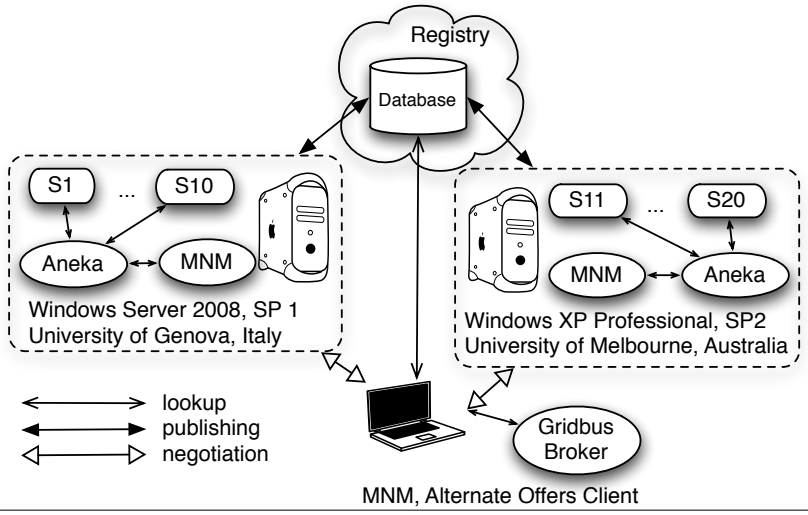

Figure 7. Testbed

of the time necessary to complete the meta-negotiation and time necessary to complete the negotiation.

Considering the fact that the time necessary to complete a meta-negotiation represents only 16.17% of the overall negotiation time, and considering the fact that we have used negotiations with only one round, we can show that the overhead of the meta-negotiations do not significantly influence the overall negotiation time.

With the presented experiments we demonstrated the applicability of our approach to the proposed architecture. Since we plan to use computational clouds in the future, the intention of the presented experiments was not to test the scalability of our approach.

6. Conclusions and Future Work

In this paper, we have presented a framework for meta-negotiations in SLA-driven Grids. Meta-negotiations bridge the gap between different requirements of service providers and consumers for conducting negotiations. We presented a meta-negotiation document through which each participant may state supported protocols, and document languages as well as the prerequisites for the starting negotiations and establishing agreements. Furthermore, we presented an case study enabling meta-negotation among providers and consumers using a registry and meta-negotiation middleware. Finally, we evaluated the meta-negotiation framework using *Aneka* and the *Gridbus* broker, and presented its results.

We plan to extend our prototype with semantic compliance of terms used within the meta-negotiation document. Thus, incorporation of semantic mappings into meta-negotiation represents a challenging future research issue. We also plan to facilitate meta-negotiation middleware with the features for the automatic generation of client software based on delivered meta-negotiations. Furthermore, we will extend our approach and incorporate it into Grid workflows.

Acknowledgments

We would like to thank Christian Vecchiola and Marcos Assuncao (University of Melbourne) for their comments on this paper. The author from the Vienna University of Technology was partially funded by the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement 215483 (S-Cube). The authors from GRIDS Lab were funded through Australian Research Council and the Dept. of Innovation, Industry and Scientific Research under Discovery Project and International Science Linkage grants respectively.

References

[1] R. J. Al-Ali, O. F. Rana, D. W. Walker, S. Jha, and S. Sohail. *G-qosm: Grid service discovery using qos properties*. Computing and Informatics, 21:363–382, 2002.

[2] J. Blythe, E. Deelman, Y. Gil. *Automatically Composed Workflows for Grid Environments*. IEEE Intelligent Systems 19(4): 16–23 2004.

[3] I. Brandic, S. Pllana, S. Benkner. *Specification, Planning, and Execution of QoS-aware Grid Workflows within the Amadeus Environment*. Concurrency and

Computation: Practice and Experience, 20(4): 331–345 John Wiley & Sons, Inc., New Jersey, March 2008.

[4] X. Chu, K. Nadiminti, Ch. Jin, S. Venugopal, and R. Buyya *Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications*. Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (e-Science 2007), Dec. 10-13, 2007, Bangalore, India.

[5] K. Czajkowski, I. Foster, C. Kesselman, V. Sander and S. Tuecke, *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*. 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh Scotland, July 2002.

[6] E. Elmroth and J. Tordsson. *A grid resource broker supporting advance reservations and benchmark-based resource selection.* in State-of-the-art in Scientific Computing, ser. LNCS. Springer- Verlag, Berlin, Germany, 2006, vol. 3732, pp. 1061-1070.Press, Los Alamitos, CA, USA, June 2004.

[7] FIPA Abstract Architecture Specification, http://www.fipa.org/specs/fipa00001/index.html

[8] I. Foster, and C. Kesselman, and G. Tsudik, and S. Tuecke. *A Security Architecture for Computational Grids*, Proc. 5th ACM Conference on Computer and Communications Security Conference, San Francisco, CA, USA, ACM Press, New York, USA, 1998.

[9] Z. Hill, J. C. Rowanhill, A. Nguyen-Tuong, G. S. Wasson, J. C. Knight, J. Basney, M. Humphrey. *Meeting virtual organization performance goals through adaptive grid reconfiguration*. 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin, Texas, USA, September 19-21, 2007.

[10] P. Iyer, A. Nagargadde, S. Gopalan, V. Sridhar. *SOA for Hybrid P2P based Self Organising Grids*. Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006), Guadeloupe, French Caribbean, 19-25 February 2006.

[11] A. Leff, J. T. Rayfield, D. M. Dias *Service-Level Agreements and Commercial Grids*. IEEE Internet Computing, vol. 07, no. 4, pp. 44-50, Jul/Aug, 2003.

[12] Z. Li, M. Parashar: An Infrastructure for Dynamic Composition of Grid Services. 7th IEEE/ACM International Conference on Grid Computing (Grid 2006), Barcelona, Spain, September 28-29, 2006.

[13] N.C. Narendra. *Generating Correct Protocols from Contracts via Commitments*. 3rd International Workshop on Service- and Process-Oriented Software Engineering (SOPOSE08), held in conjunction with the 2008 IEEE International Conference on Services Computing (SCC 2008), Honolulu, Hawaii, USA, July 8-11, 2008.

[14] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar. *A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing.* in Proceedings of the 2005 European Grid Computing Conference (EGC 2005), Amsterdam, The Netherlands, February, 2005.

[15] D. M. Quan, J. Altmann: Mapping of SLA-based Workflows with light Communication onto Grid Resources. Grid Service Engineering and Management: The 4th International Conference on Grid Service Engineering and Management, GSEM 2007, September 25-26, 2007, Leipzig, Germany.

[16] D. Thain, T. Tannenbaum, and M. Livny. *Distributed Computing in Practice: The Condor Experience*. Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

[17] M. Wieczorek, S. Podlipnig, R. Prodan, T Fahringer. *Bi-criteria Scheduling of Scientific Workflows for the Grid Cluster Computing and the Grid*. IEEE International Symposium on Cluster Computing and the Grid, Lyon, France, 19-22 May 2008.

[18] W. Vambenepe, C. Thompson, V. Talwar, S. Rafaeli, B. Murray, D. S. Milojicic, S. Iyer, K. I. Farkas, M. F. Arlitt. *Dealing with Scale and Adaptation of Global Web Services Management*. International Journal of Web Services Research, 4(3): 65-84, 2007.

[19] S. Venugopal, R. Buyya and L. Winton, *A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids*, Concurrency and Computation: Practice and Experience, 18(6): 685-699, Wiley Press, New York, USA, May 2006.

[20] S. Venugopal, X. Chu, R. Buyya. *A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol*. 16th International Workshop on Quality of Service (IWQoS 2008), June 2-4, 2008, University of Twente, Enschede, The Netherlands.

[21] L.-H. Vu, F. Porto, K. Aberer, M. Hauswirth. *An Extensible and Personalized Approach to QoS-enabled Service Discovery.*, Eleventh International Database Engineering and Applications Symposium (IDEAS 2007), Banff, Alberta, Canada, September 6-8, 2007.

[22] Web Service Level Agreement (WSLA), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[23] WS-Agreement, http://www.ogf.org/documents/GFD.107.pdf

[24] K. Xiong, H. Perros. *SLA-based service composition in enterprise computing*. IEEE International Workshop on Quality of Service, Enschede, The Netherlands, 2008.