A SLA-based Framework with Support for Meta-scheduling in Advance for Grids

Javier Conejero[1], **Blanca Caminero**[1], **and Carmen Carrión**[1]
[1]Albacete Research Institute of Informatics(I^3A)
University of Castilla–La Mancha. Campus Universitario, 02071, Albacete. Spain

Abstract— *Quality of Service (QoS) is one of the most important and active research topics within Grid technology. But the emerging transformation from a product oriented economy to a service oriented economy establishes a new scenario where actual QoS mechanisms need to be enforced and new QoS mechanisms needed. Service Level Agreements (SLAs) are considered the cornerstone born to fulfill those requirements and the key concept that boosts the Grid economic exploitation. Therefore mechanisms to negotiate and manage SLAs become necessary.*

The aim of this paper is to address QoS within Grids by proposing an architecture capable of providing a service level agreement negotiation service and management for the agreed terms.

Our proposal introduces two new layers on the top of a Grid architecture with scheduling in advance feature, responsible of the SLAs handling and management. These layers consider the SLA standard proposed by the Open Grid Forum (OGF). In addition, they can also handle meta-scheduling mechanisms for non trivial execution parameters (e.g. economical, energetic, etc.).

Resulting as a potential improvement over QoS that SLAs in coordination with scheduling in advance can achieve within Grids. It is also shown the flexibility of this architecture and how it can be easily adapted to work over different Grid middlewares improving the interoperability of heterogeneous Grids.

Keywords: Service Level Agreement, Quality of Service, Grid Computing, WS-Agreement, Scheduling in advance.

1. Introduction

With the emerging interest on Cloud Computing and the new paradigm that it has introduced into Distributed Computing, QoS and *pay–per–use* models, the economy is transforming from a product oriented economy to a service oriented economy. This trend is boosting the economic exploitation of Distributed Computing environments like Grid Computing and High Performance Computing [1]. Therefore, new mechanisms are needed in order to evolve and adapt to the new needs.

Grids, Clusters and their combinations have been rediscovered with this new trend because of the business interest on those technologies. Enterprises are interested on exploiting the resources they own in order to get benefits. This pushes research to go on this direction and Service Level Agreements (SLAs) and their management are intended to fulfill these needs [2].

The exploitation of these technologies is defined by enterprises and it can be economic, time scheduled or related to other terms. So an agreement is needed between the two parts involved: user and provider. Thus, the negotiation and enforcement to fulfill the terms defined within a SLA are directly related with the QoS the user expects to receive.

The SLA concept within Grid environments can be defined as a contract between a user and a Grid service provider in which participants expectations and obligations are explicitly defined [3].

So, it can be said that SLAs represent a contract between the user and the Grid Service Provider where the QoS expected to be received from the Grid service provider and the legal implications are explicitly exposed.

The scope of this paper is to improve the QoS within Grids by proposing an architecture capable of providing a framework to support Service Level Agreements, negotiation and management of the agreed terms included on them.

The structure of this article is as follows: general concepts about SLAs are described in Section 2. The main problems in SLAs the proposed architecture is expecting to solve are pointed out in Section 3. Section 4 presents the architecture objective of this paper in detail. It also contains three subsections, the first two for the new layers proposed: SLA-Manager (4.1) and SLA-Backend (4.1); and the last one describing the underlying technologies, background and middleware (4.3). Related work is presented in Section 5. Finally, Section 6 concludes the paper and describes the guidelines for future work.

2. SLAs general concepts

The SLAs state of the art is mainly addressed by the WS-Agreement specification proposed by the Open Grid Forum (OGF) [4]. Previous work on this field was done by WSLA [5] and SLAng [6] specifications, but they are nowadays unmaintained. Due to enterprises behind the OGF interested on SLAs, that participate on its specification, it is defined as the most important these days; and consequently the actual standard.

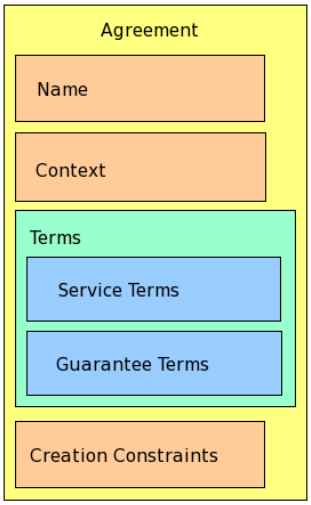

Fig. 1: SLA structure

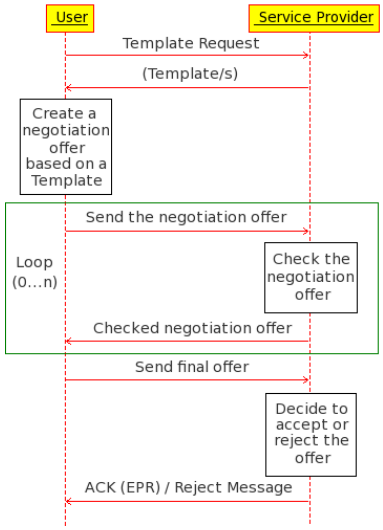

Fig. 2: WS-Agreement negotiation protocol with extension

WS-Agreement has been widely adopted by most projects where SLAs are on their roadmap [7], [8], [9], [10], [11], [12] but due to its limitations, alternatives have been developed causing the contrary effect of the definition of standard. Each project attempts to implant SLAs following the WS-Agreement specification adapting it for their private purposes. This causes, in some cases, incompatibility between them.

WS-Agreement is a Web Service protocol developed by GRAAP-WG from the OGF. It defines explicitly the structure of the SLA, a web service protocol for the SLA establishment between users and service providers, and the templates to make service provider discovery easier. Moreover, it defines the constraints, life-cycle, monitoring and runtime states for SLAs [4].

The structure defined by WS-Agreement for SLAs (Figure 1) [4] consists of four main blocks. The *Name* block only contains the name of the agreement which is optional and it is not the agreement identifier (just for human identification). The *Context* block contains all agreement details related to the context of the agreement (e.g. client id, provider id, etc.). These details are mandatory, but the specification lets the service provider extend them with new ones. The *Terms* block contains all terms related to assurances and commitments between user and service provider. The *Service Terms* are usually known as service descriptors, because they reference services or service terms properties (e.g. Number of CPUs, Amount of RAM, etc.). The *Guarantee Terms* specify the value or the QoS expected for the terms specified on the service terms (e.g. 4 (CPUs), 2 (RAM Gb), etc.). Finally, in the *Creation Constraints* block, the user can get some information from the service provider in order to know the limitations of the Service terms field. This block only appears on the negotiation template.

Finally, the negotiation protocol defined by the WS-Agreement specification and updated by the WS-Agreement extension [13] (Figure 2) shows how users and service providers interact in order to negotiate an agreement defined by an SLA. It specifies a negotiation protocol with renegotiation support.

3. SLAs and QoS

The main issues to be tackled in the process of integration of SLAs on a distributed environment, in order to provide QoS guarantees, are mainly two: a) the extraction/choice of the QoS properties provided by the contract and b) to adjust the behaviour of the system according to them.

One of the main problems related to the first issue that appears when trying to integrate SLAs on a distributed environment are the temporal restrictions (e.g. start-time, deadline, etc. of jobs) that can be defined on an agreement. This problem is not tackled by the WS-Agreement specification and it is supposed to be addressed by other mechanisms.

Nowadays, some distributed systems have reservation in advance capability. This can solve the problem of temporal restrictions but due to the fact that reservations are not always possible, our proposal is going to be based on meta-scheduling in advance in Grid environments.

Meta-scheduling in advance can be defined as the first step of a reservation in advance process. In consequence, only job execution time periods and resources are selected, keeping track of them besides the resources status, but without making any physical reservation [14].

QoS in terms of temporal restrictions has been actively researched, mainly because metrics where known in advance (e.g. start-time) or could be evaluated from a profiling process (e.g. duration). And, as a consequence, job scheduling

based on these metrics can be highly optimized in reservation or scheduling in advance algorithms.

Furthermore, the integration of SLAs in Grid environments can be useful due to all the information contained on them. This information is obviated by most meta-schedulers (e.g. Condor [15], UNICORE [16], gLite [17], etc.) where only temporal metrics are used, preventing Grid environments from exploiting this information and, as a consequence, limiting the QoS reached.

But the use of other metrics, represented as terms on an agreement, could be used to take decisions that result in an improvement of QoS in terms of usage (e.g. number of CPUs needed, amount of RAM requested, etc.) or in terms of energy saving among others. Energy saving represents a highly active research topic within GreenIT [18]. In other words, these metrics could be used for a smarter job meta-scheduling system, managed by specific provider objectives.

This can not be managed by actual schedulers but it represents a very important source of information for the meta-scheduling process. So, the adjustment of the behaviour of the system according to them is needed in order to improve the QoS, as cited on the second issue.

Our proposal is presented to tackle these two problems by using both: a) a meta-scheduling in advance layer and b) a high level term meta-scheduling layer; such as will be detailed in next section.

4. Proposed Architecture

The following section shows the architecture proposed to solve the problems highlighted in Section 3.

Studying in detail the problems related with the adoption of SLAs in Grid Computing it can be thought that the SLA management, and by consequence all WS-Agreement specification, should be implemented into the middleware of the Grid. This could be a solution if the middleware has native support for reservation in advance or scheduling in advance as shown in Section 3. This is not the case in most Grid middlewares, so native support for SLAs and their management is not available on them. Therefore we propose to exploit the advantages given by the GridWay meta-scheduler enhanced with Scheduling in Advance Layer (SA-Layer) over Globus Toolkit 4 (GT4), which enables the scheduling in advance capability.

Our proposal consists of two new layers: SLA-Manager and SLA-Backend; solution which places on top of GT4 with the GridWay meta-scheduler and the SA-Layer (Figure 3) to handle SLAs on the Grid with scheduling in advance capability.

This solution has been designed to isolate the problems produced by the WS-Agreement specification and evolution (SLA-Manager), and it also isolates the problem of how to manage time limitations terms from others if we want them to participate on the meta-scheduling process (SLA-Backend).

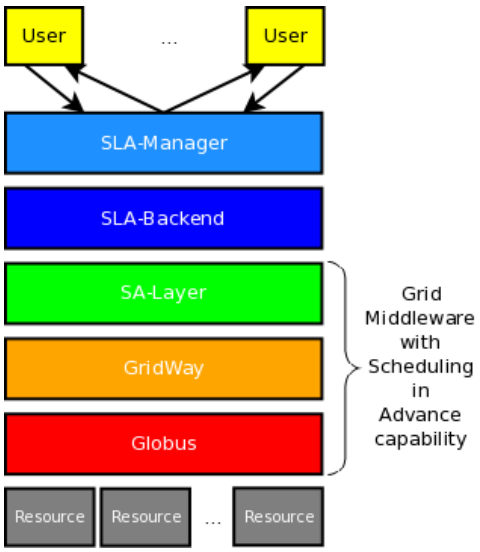

Fig. 3: Proposed Architecture

Furthermore, this solution has been designed for a not invasive implantation over the actual Grids and their structure if based on GT4. In other words, working Grids will not need to change anything from their actual configuration in order to extend their functionalities with WS-Agreement, just set the new layers in top of GT4. Service providers will be able to do a stagger transition of their services because with this proposal WS-Agreement will be working independently from its implantation.

To take advantage of the functionalities and libraries provided by the middleware, the SLA-Manager and SLA-Backend layers should be implemented as Web Services. This will make easier the security handling and, by defining well known interfaces, the use of this service from other web services (Figure 4).

Although the SA-Layer is external to GT4, the SLA-Backend must be able to interact with it and take advantage of it. GridWay is needed just to support SA-Layer requirements. The two layers proposed do not need interaction with

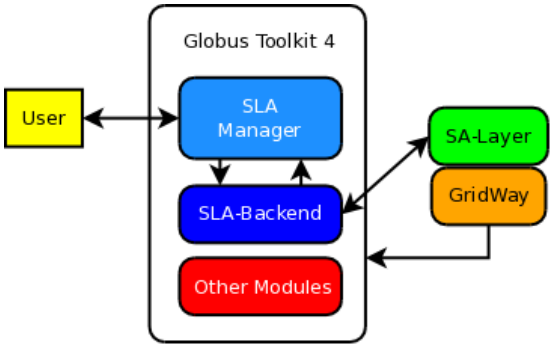

Fig. 4: Software design

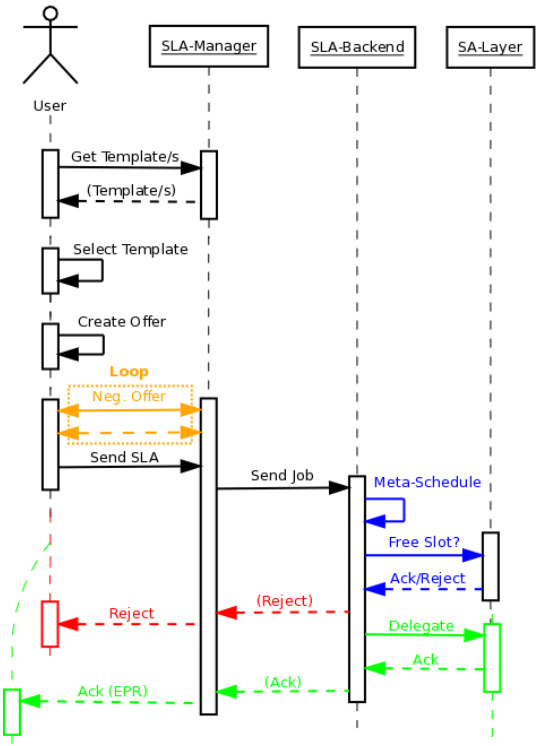

Fig. 5: Basic sequence diagram

it. If in some case, another meta-scheduler is desired to be used in place of GridWay and SA-Layer, the isolation is guaranteed and the only modification should be on the interfaces (if needed).

The workflow of our proposed architecture is shown in Figure 5. The user is who starts the negotiation process (as specified in WS-Agreement) by requesting a template. The user interacts with the SLA-Manager and performs the sequence diagram specified in WS-Agreement as shown in Figure 2. During this process the SLA-Manager interacts with the SLA-Backend in order to decide if the request done by the user is going to be accepted or rejected. If the request is rejected, the SLA-Backend layer communicates the decision to the SLA-Manager layer, which sends a reject message to the user. On the contrary, if the request is accepted, the SLA-Backend layer delegates the execution of the job on the SA-Layer and sends an acknowledgement message (ACK) with the corresponding endpoint reference (EPR).

On the next subsections, the building blocks of the proposed architecture will be described in detail.

4.1 SLA-Manager

The SLA-Manager layer has the mission of interacting with the user, offering the mechanisms to negotiate an agreement following the WS-Agreement specification.

It is designed to be implemented as a GT4 Web Service (Figure 4) so it can take advantage of the Web Services functionality (e.g. discovery, interfaces, etc.) and security. Security is essential when negotiating an agreement (e.g. trusted user?). So deploy the SLA-Manager as a GT4 Web Service can ensure security due to the Globus Security Infrastructure (GSI).

For this proposal, the SLA-Manager follows the WS-Agreement specification, but it can be easily extended for other purposes or extensions of the specification.

4.2 SLA-Backend

The SLA-Backend layer is defined as the intermediate layer between the SLA-Manager and the middleware infrastructure. Its main mission is to decide what to do with the requests performed by users.

These decisions are intended to be taken within this layer, so intelligence must be implemented in order to meta-schedule the incoming jobs.

This layer isolates the traditional scheduling decisions made by the middleware scheduler, or in our architecture, the SA-Layer meta-scheduler, which usually use time related parameters to take decisions.

The SLA-Backend sets a new higher level meta-scheduler that delegates time related parameters to be used on a lower level meta-scheduler (SA-Layer in our proposal), and it is thought to use any other (or more than one) metric/parameter like energy consumption, economy or any other term defined on a SLA.

So the SLA-Backend takes higher level decisions, letting the lower layers decide what to do with the jobs in terms of time-scheduling. This lets the service provider to have more control over the whole infrastructure in terms of human understandable metrics or parameters.

This module is also designed to be a GT4 Web Service to ensure security and to make the interconnection with the SLA-Manager easier (Figure 4).

4.3 SA-Layer and Middleware

Scheduling in advance has been chosen as the solution for the temporary restriction imposed by the use of SLAs in Grid environments. The reasons for this decision are mainly two: some kind of 'in advance' task is needed in order to solve the temporary restrictions (but scheduling in advance has been chosen instead of reservation in advance because reservations are not always possible); negotiation and renegotiation of SLAs can be extremely useful into scheduling and rescheduling in advance due to its not invasive reservation of resources. So, a middleware with this capability will represent the principal pillar in our proposal.

On a first approach, Globus Toolkit [19] middleware was chosen due to its "de facto" consideration. But its lack of scheduling in advance capability made us consider

another approach: use a solution that could give scheduling in advance capability to Globus Toolkit.

For our purpose, we are going to use the architecture presented on [14], [20] as base/middleware. On these works a layered architecture is proposed based on the use of Globus Toolkit 4, GridWay and the Scheduling in Advance layer (SA-Layer) innovation. This lets the system make the most of scheduling in advance capability.

The SA-Layer [14], [20] is the cornerstone of it. Using GridWay [21] functionalities like resource discovery and monitoring, work submitting and execution monitorization among others. The SA-Layer enables the scheduling in advance capability.

Moreover, the fact of being an architecture that respects the actual evolution of each module, isolating the tasks and responsibilities, makes this architecture the perfect base for our design.

On the contrary, this solution adds complexity to our approach, but as seen in this section, this does not mean a disadvantage.

This stack can be replaced by another one with scheduling in advance capability but it is necessary that it respects the interfaces and libraries defined on GT4 and SA-Layer interfaces for a swap without problems. Although wrappers could be designed in order to allow and facilitate this feature.

5. Related Work

Due to the importance of the topic exposed on this article, many efforts have been done on SLAs. For example, on their management [22], [23], QoS implications [24], semantic and virtualization exploitation [25] and specially on its standardization. The *Open Grid Forum* (OGF) proposes the use of WS-Agreement (*Web Services Agreement Specification*) [4], which is nowadays the most important and widely adopted.

Not only theoretical work has been done on this topic. Many projects have been interested on introducing SLA negotiation and management mechanisms on real Grids. Some of them will be analysed on the next paragraphs, focusing on their contribution to the topic.

AssessGrid (*Advanced Risk Assessment and Management for Trustable Grids*) [7] is a project that produced a free generic, configurable, trustable and interoperable software for the risk assessment, their management and decision support for Grid environments. It is freely available on the AssessGrid project web [7].

This project relies on SLAs. They are one of the most important aspects of this project, giving great importance to its management and negotiation. Its implementation is WS-Agreement based and developed for Globus Toolkit 4.

The AssessGrid project introduces a new innovation on SLAs field. The concept of a third entity into negotiation workflow: the broker. The user interaction with this entity is not mandatory, because if it were, they would not be able to talk about a WS-Agreement specification based implementation. Moreover, it introduces new improvements to it: provider selection based on the best negotiation results, negotiation mediator (letting the broker distribute the incoming tasks specified on a SLA through the available service provider) and as a real negotiator (letting the broker negotiate instead of the user in order to get the best agreement) [26].

On this project, the broker service is in charge of solving the problem of job meta-scheduling and delegation. But as seen before, its mission is oriented to risk assessment.

It is important to mention that the negotiation protocol incorporated on this project respects the WS-Agreement specification, but it contains some other techniques that let them introduce the renegotiation concept and new negotiation workflows beyond the WS-Agreement negotiation extension [27].

Another interesting project at present is SLA4D-Grid (*Service Level Agreements for D-Grid*)[28], [9]. This project is designing and realising a SLA layer for the D-Grid (*Germany's National Grid*).

This project is based on the WS-Agreement specification. Furthermore, it is prepared to be easily extended for specific business purposes. It is implemented for Globus Toolkit 4 but designed for different middlewares (UNICORE [16], Globus [19] and gLite [17]) and services present on D-Grid.

Moreover, it has already produced a module known as Negotiation Manager which can offer WS-Agreement functionalities for Globus Toolkit 4 [29]. As expected, this project experienced the same problem exposed on this article. They solved it by using the ZIBARS module from AstroGrid-D project [30]. By this way they can handle scheduled jobs by delegating them to Reservation in Advance software (Maui meta-scheduler). Please note that this project does not support desktop Grids unlike our proposal.

WSAG4J (*WS-AGreement for Java*) [12] is a generic WS-Agreement specification framework developed by the Fraunhofer Institute (SCAI). It provides a quick and easy environment for the development of WS-Agreement based services. Moreover, it facilitates application and service debugging before their deployment.

It ensures that all application and services developed using this framework can be directly deployed into other environments if these respect the WS-Agreement specification.

Unlike AssessGrid and SLA4D-Grid projects, WSAG4J is implemented to work as an Apache Tomcat service. So, it does not let the user interact with a real Grid environment and it constitutes just a development framework.

Other projects like: Brein [8] and SmartLM [11] among others propose their own implementations of the WS-Agreement specification, but most of them modify it for their specific purposes that do not interfere with the basic operation. Thus possibly compromising the interoperability between Grids.

6. Conclusions and Future Work

This article presents a five layer architecture to empower the use of SLAs in Grid environments and solve the problems of job scheduling with advance timing requirements.

Moreover, it shows an architecture managed by a policy where SLA terms are on a higher level than execution parameters. So, specific schedulers can coexist in order to improve the QoS perceived by the user. In addition, new algorithms for these new high level term scheduler can now be designed and evaluated.

Furthermore, this proposal goes far away by not modifying the middleware and isolating the responsibilities in layers that can be easily adapted for private or business purposes.

This proposal pushes Grid environments to move on a more flexible management way where time restrictions not related parameters are involved in the job scheduling process. This help business exploitation of Grid resources by terms of economy and energy among others.

Some related work is analysed and results in a lack of a generic proposal that can handle SLAs where the scheduling in advance capability is available.

Future work steps are: implementation of the architecture proposed on this this article on a real Grid, and the design of tests to get empirical results regarding performance and usability.

Acknowledgement

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants "CSD2006-00046", "TIN2009-14475-C04" and through a FPI scholarship asociated to "TIN2009-14475-C04-03" project. It was also partly supported by JCCM under Grant "PII1C09-0101-9476".

References

[1] D. Armstrong and K. Djemame, "Towards Quality of Service in the Cloud," in *Proc. of the 25th UK Performance Engineering Workshop*, Leeds, UK., 2009.

[2] V. Stantchev and C. Schröpfer, "Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing," in *Proc. of the 4th International Conference on Advances in Grid and Pervasive Computing (GPC)*, Geneva, Switzerland, 2009.

[3] J. Padgett, K. Djemame, and P. Dew, "Grid-Based SLA Management," in *Proc. of the European Grid Conference (EGC'2005)*, Amsterdam, The Netherlands, 2005.

[4] A. A. et al, "Web Services Agreement Specification (WS-Agreement) GFD-R-P.107," Tech. Rep., March 2007, https://forge.gridforum.org/projects/graap-wg/.

[5] "WSLA: Web Service Level Agreements," [Online]. Available: http://www.research.ibm.com/wsla/, Date of last access: 29th January, 2011.

[6] D. D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," in *Proc. of the International Workshop of Future Trends of Distributed Computing Systems*, Los Alamitos, USA, 2003.

[7] "AssessGrid," [Online]. Available: http://www.assessgrid.eu, Date of last access: 24th January, 2011).

[8] EU-Brein - Bussiness Objective driven REliable and Intelligen grids for real busiNess, [Online]. Available: http://www.eu-brein.com/, Date of last access: 24th January, 2011.

[9] "D-Grid - SLA4D-Grid - Service Level Agreement für das D-Grid," [Online]. Available: http://www.d-grid-ggmbh.de/index.php?id=89, Date of last access: 27th January, 2011.

[10] "SLA at SOI," [Online]. Available: http://sla-at-soi.eu/, Date of last access: 28th February, 2011.

[11] "SmartLM," [Online]. Available: http://www.smartlm.eu, Date of last access: 25th January, 2011.

[12] "WSAG4J - WS-Agreement framework for Java," [Online]. Available: http://packcs-e0.scai.fraunhofer.de/wsag4j/, Date of last access: 27th January, 2011.

[13] O. W. et al, "WS-Agreement Negotiation Version 1.0," Tech. Rep., January 2011.

[14] L. T. et al, "Network-aware meta-scheduling in advance with autonomous self-tuning system," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 486 – 497, 2011.

[15] "Condor Project: High Throughput Computing," [Online]. Available: http://www.cs.wisc.edu/condor/, Date of last access: 4th February, 2011.

[16] "UNICORE: Distributed Computing and Data Resources," [Online]. Available: http://www.research.ibm.com/wsla/, Date of last access: 4th February, 2011.

[17] "gLite: Lightweight Middleware for Grid Computing," [Online]. Available: http://glite.cern.ch/, Date of last access: 4th February, 2011.

[18] G. Laszewski and L. Wang, "GreenIT Service Level Agreements," in *Grids and Service-Oriented Architectures for Service Level Agreements*, P. Wieder, R. Yahyapour, and W. Ziegler, Eds. Springer US, 2010, pp. 77–88. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-7320-7_8

[19] I. T. Foster, *Globus Toolkit Version 4: Software for Service-Oriented Systems.*, ser. Lecture Notes in Computer Science, H. Jin, D. A. Reed, and W. Jiang, Eds. Springer, 2005, vol. 3779.

[20] L. T. et al, "Meta-Scheduling in Advance using Red-Black Trees in Heterogeneous Grids," in *Proc. of the High Performance Grid Computing Workshop (HPGC), hold jointly with the International Parallel & Distributed Processing Symposium (IPDPS)*, Atlanta, USA, 2010.

[21] C. V. et al, "Federation of teragrid, egee and osg infrastructures through a metascheduler," *Future Generation Computing Systems*, vol. 26, pp. 979–985, July 2010. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.04.004

[22] J. P. et al, "Predictive Adaptation for Service Level Agreements on the Grid," *International Journal of Simulation Systems, Science and Technology*, vol. 7, no. 2, pp. 29–42, March 2006.

[23] W. Theilmann and L. Baresi, "Multi-level SLAs for Harmonized Management in the Future Internet," *Towards the Future Internet*, pp. 193–202, 2009.

[24] I. B. et al, "Advanced QoS Methods for Grid Workflows Based on Meta-Negotiations and SLA-Mappings," in *Proc. of the 3rd Workshop on Work ows in Support of Large-Scale Science. In conjunction with Supercomputing*, Austin, USA, 2008.

[25] J. E. et al, "Exploiting semantics and virtualization for SLA-driven resource allocation in service providers," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 5, pp. 541–572, April 2010.

[26] M. P. et al, "A Comparison of SLA Use in Six of the European Commissions FP6 Projects," Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, Tech. Rep. TR-0129, April 2008.

[27] W. Ziegler, P. Wieder, and D. Battré, "Extending WS-Agreement for dynamic negotiation of Service Level Agreements," Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, Tech. Rep. TR-0172, August 2008.

[28] "SLA4D-Grid Negotiation Manager," [Online]. Available: http://www.sla4d-grid.de/, Date of last access: 27th January, 2011.

[29] M. Raack, "Documentation of the SLA4D-Grid Negotiation Manager (Globus)," Tech. Rep., July 2010, http://www.sla4d-grid.de/sites/default/files/SLA4D-Grid_Negotiation-Manager.pdf.

[30] T. Röblitz, "Deliverable 5.6 -Resource Management for Grid-Jobs," Tech. Rep., May 2009.