European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Considerations for negotiation and monitoring of Service Level Agreements

Wolfgang Ziegler and Oliver Wäldrich
{Wolfgang.Ziegler, Oliver.Waeldrich}@scai.fraunhofer.de
Fraunhofer Institute SCAI, Department of Bioinformatics
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Philipp Wieder
philipp.wieder@udo.edu
Technical University Dortmund
Dortmund, Germany

Toshiyuki Nakata
t-nakata@cw.jp.nec.com
Central Research Lab. NEC
1753, Shimonumabe, Nakahara-Ku, Kawasaki,Kanagawa 211-8666, Japan

Michael Parkin
mparkin@bsc.es, CoreGRID fellow
ERCIM
2004 Route des Lucioles, BP 93 06902 Sophia Antipolis, France

CoreGRID Technical Report
Number TR-0167
July 30, 2008

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.eu

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Considerations for negotiation and monitoring of Service Level Agreements

Wolfgang Ziegler and Oliver Wäldrich
{Wolfgang.Ziegler, Oliver.Waeldrich}@scai.fraunhofer.de
Fraunhofer Institute SCAI, Department of Bioinformatics
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Philipp Wieder
philipp.wieder@udo.edu
Technical University Dortmund
Dortmund, Germany

Toshiyuki Nakata
t-nakata@cw.jp.nec.com
Central Research Lab. NEC
1753, Shimonumabe, Nakahara-Ku, Kawasaki,Kanagawa 211-8666, Japan

Michael Parkin
mparkin@bsc.es, CoreGRID fellow
ERCIM
2004 Route des Lucioles, BP 93 06902 Sophia Antipolis, France

CoreGRID TR-0167

July 30, 2008

Abstract

Service Level Agreements (SLA) may be used to establish agreements on the quality of a service between a service provider and a service consumer. This report gives an overview on current work done on negotiation and re-negotiation of SLAs within the CoreGRID Institute on Resource Management and Scheduling and in the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG) of the Open Grid Forum (OGF). An introduction is given on the proposed OGF recommendation WS-Agreement and its state, followed by a presentation of use-cases for negotiation stemming from from implementations of WS-Agreement. We then describe the approaches currently under discussion in the GRAAP-WG to be consolidated to a negotiation protocol for Service Level Agreements using WS-Agreement. Since the working group of the OGF has not yet consolidated the approaches presented there is not yet a single proposal for a negotiation protocol. However, during the last meeting of the working group the members identified the three approaches that will now be used as a base to specify the protocol for negotiation and re-negotiation of SLAs. Furthermore, the report discusses built-in functionality of WS-Agreement and additional requirements for the monitoring of SLAs.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

1 Introduction

Service Level Agreements (SLA) may be used to establish agreements on the quality of a service between a service provider and a service consumer. Today these service level agreements are often created manually for individual services or are established through a framework contract between service provider and customer. Given the dynamic nature of the requirements towards service provisioning in Grids and the increased flexibility in selecting the most appropriate service provider for a required service new mechanisms are needed to create Service Level Agreements on the fly and to modify at a later state.

Inspired by previous developments in the web-service domain research and development in the Grid domain started addressing electronic Service Level Agreements on the quality or properties of services or resources to be used to solve a problem. The negotiation of the agreement between two parties is usually done automatically by appropriate software instances of the service provider and service consumer. In this overview we focus on WS-Agreement, which is the proposed recommendation of the Open Grid Forum (OGF) to create and monitor Service Level Agreements. It has been specified by the Grid Resource Allocation Agreement Working Group (GRAAP-WG) where CoreGRID researchers have been driving the development together with industrial partners from NEC, IBM, HP and Platform since 2002. The engagement in the GRAAP-WG directly relates to the Institute's Task 8 - Service Level Agreements of the CoreGRID Institute on Resource Management and Scheduling (RMS).

Following the creation of an SLA with WS-Agreement both parties of the agreement need access to the state information of the agreement. In the case of WS-Agreement this information may be retrieved through functions provided by the Web Service Resource Framework (WSRF) [24] on which the Web service expression of the WS-Agreement specification is based. Monitoring a SLA also empowers both parties to detect potential violations of the agreement and to take appropriate measures.

Negotiation and Monitoring of SLAs is of particular importance for commercial service providers. Standards are required to guarantee interoperability and the GRAAP-WG has become the focal point of these efforts, which is another reason for CoreGRID to heavily engage in this activity.

The WS-Agreement specification version 1.0 already includes a protocol for negotiation of agreements. However, this protocol was designed to cover the most simple and general case: an offer for an SLA is made by either of the two parties and the respective other party may accept or reject the offer. No further negotiation, e.g. in form of a counter offer or request for modification is supported. Since there are situations where an agreement has to be modified at a later state or the process of creating an agreement needs more than the single step approach described above a more sophisticated protocol for negotiation and re-negotiation is needed in addition to the existing one.

One objective of the current work of the Grid Resource Allocation Agreement Group is producing a specification for an interoperable protocol for sophisticated negotiations and re-negotiation of SLAs.

In Section 2 we give a brief overview on WS-Agreement. In Section 3 we describe some use-cases for negotiation and re-negotiation of Service Level Agreements, the requirements for a protocol, and finally, the three approaches addressing different requirements, which have been selected for consolidation to form a single protocol during the last OGF meeting:

- A Contract Re-negotiation Protocol
- Dynamic SLA negotiation based on WS-Agreement
- Protocol considerations based on the current WS-Agreement specification

Section 4 addresses issues related to monitoring of agreement, and section 5 includes a brief outlook on the ongoing development concludes the report.

2 Overview on WS-Agreement and its state

Service Level Agreements have a certain life cycle, which they are running through from the initial preparation of the templates for an agreement up the evaluation whether the agreement has been fulfilled, or partly or completely violated. W. Sun et al [8] describe the SLA life cycle consisting of five phases; these phases are shown in Figure 1

The activities in the individual phases can be described as follows:

- SLA Development: In this phase the SLA templates are developed.

Figure 1: SLA lice-cycle.

- Negotiation: In this phase the SLA is negotiated and the contracts are executed.
- mplementation: where the SLA is generated.
- Execution: The SLA is executed, monitored, and maintained.
- Assessment: Evaluation of the SLA performance. In this phase, a re-evaluation of the initial SLA template might be done.

In contrast to the linear sequence of phases described above additional transitions between the Implementation or the Execution phase back to the Negotiation phase will be performed when re-negotiation of an agreement is required.

In this proposal we are focussing on the Negotiation phase and the Execution phase (as far as monitoring of agreements is concerned). For the implementation of SLAs we consider WS-Agreement [2], which is described briefly in the following paragraphs.

Web-Services Agreement (WS-Agreement) is a specification defining a language and a protocol to create Service Level Agreements (SLA) between a service provider and a service consumer (usually a customer of the provider). WS-Agreement currently is used in different projects, domains and on different levels to establish electronic agreements on the quality of a service (QoS) between a service provider and a service consumer.

The WS-Agreement specification version 1.0 already includes a protocol for negotiation of agreements. However, this protocol was designed to cover the most simple and general case: an offer for an SLA is made by either of the two parties and the respective other party may accept or reject the offer. No further negotiation, e.g. in form of a counter offer or request for modification is supported. Since there are situations where an agreement has to be modified at a later state or the process of creating an agreement needs more than the single step approach described above a more sophisticated protocol for negotiation is needed in addition to the existing one.

WS-Agreement has been specified by the Open Grid Forum (OGF) [17] Grid Resource Allocation Agreement Working Group (GRAAP-WG) [9] and is a proposed OGF recommendation since May 2007. Currently members of the GRAAP-WG are working on the document describing the experience made during the interoperability and compliance tests. With this document the Grid Forum Steering Group of the OGF will be able to decide on the transition of the WS-Agreement specification from a proposed recommendation to a recommendation.

2.1 WS-Agreement structure and terminology

The purpose of Web Services Agreement Specification is supporting establishing an agreement on the usage of Web Services between a service provider and a consumer. WS-Agreement defines a language and a protocol to represent the services of providers, create agreements based on offers and monitor agreement compliance at runtime. Roles of actors in WS-Agreement and the layered service model of WS-Agreement are depicted in Figure 2. WS-Agreement allows advertising the capabilities of service providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime.

An agreement defines a relationship between two parties that is dynamically established and dynamically managed. The objective of this relationship is to deliver a service by one of the parties. In the agreement each party agrees on the respective roles, rights and obligations.

A provider in an agreement offers a service according to conditions described in the agreement. A consumer enters into an agreement with the intent of obtaining guarantees on the availability of one or more services from the provider. Agreements can also be negotiated by entities acting on behalf the provider and / or the consumer. An agreement creation process usually consists of three steps:

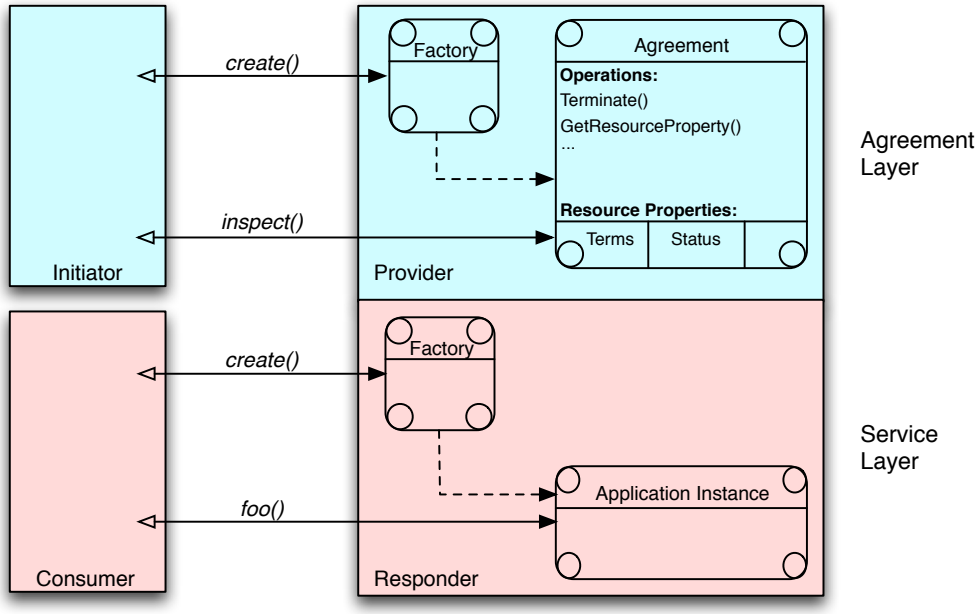

Figure 2: WS-Agreement Conceptual Layered Service Model. Note: The names of the different operations and attributes are not normative, nor is the assignment of initiator and responder roles to service consumer and provider.

- The initiator retrieves a template from the responder, which advertises the types of offers the responder is willing to accept.
- The initiator then makes an offer,
- Which is either accepted or rejected by the responder.

WS-Agreement-Negotiation sitting on top of WS-Agreement, furthermore describes the re/negotiation of agreements.

An agreement consists of the agreement name, its context and the agreement terms. The context contains information about the involved parties and metadata such as the duration of the agreement. Agreement terms define the content of an agreement: Service Description Terms (SDTs) define the functionality that is delivered under an agreement. A SDT includes a domain-specific description of the offered or required functionality (the service itself). Guarantee Terms define assurance on service quality of the service described by the SDTs. They define Service Level Objectives (SLOs), which describe the quality of service aspects of the service that have to be fulfilled by the provider. The Web Services Agreement Specification allows the usage of any domain specific or standard condition expression language to define SLOs. The specification of domain-specific term languages is explicitly left open. Figure 3 shows the structure of an Agreement.

3 Approaches for a (re-)negotiation protocol

The first part of this section presents some generic use-cases followed by some concrete implementations of these use cases. The second part includes requirements for a protocol suitable for negotiation and re-negotiation gathered over the last year in the context of the GRAAP-WG. In the third part the three approaches currently investigated in the GRAAP-WG will be introduced and discussed.

3.1 Use cases for negotiation

3.1.1 Agreement on multiple QoS Parameters

In an environment consisting of several clusters operated in different administrative domains SLAs might be used for co-allocation or the resource allocation for workflows or distributed applications. A typical use-case is the co-

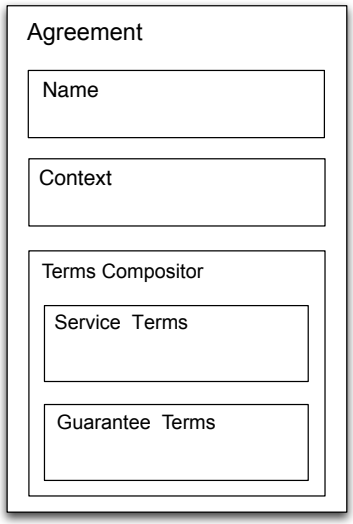

Figure 3: Structure of an agreement.

allocation of multiple compute resources together with the network links between these resources with a dedicated QoS to run a distributed parallel application. The user specifies his request and the resource orchestrator starts the negotiation with the local scheduling systems of the compute resources and the with the network resource management system (NRMS) in order to find a suitable time-slot, where all required resources are available at the same time. Once a common time-slot is identified the orchestrator requires the reservation of the individual resources. Again, this reservation can be expressed as a Quality of Service and an SLA may be created where the reservation is fixed is a binding agreement for the two parties.

During the first three years of CoreGRID the German VIOLA project citeviola-project was running in parallel, addressing reservation and co-allocation of multiple resources across different administrative domains. Two CoreGRID partners contributed to the developments of this project and were active in the RMS Institute at the same time, which led to a strong integration of this national R&D project with CoreGRID.

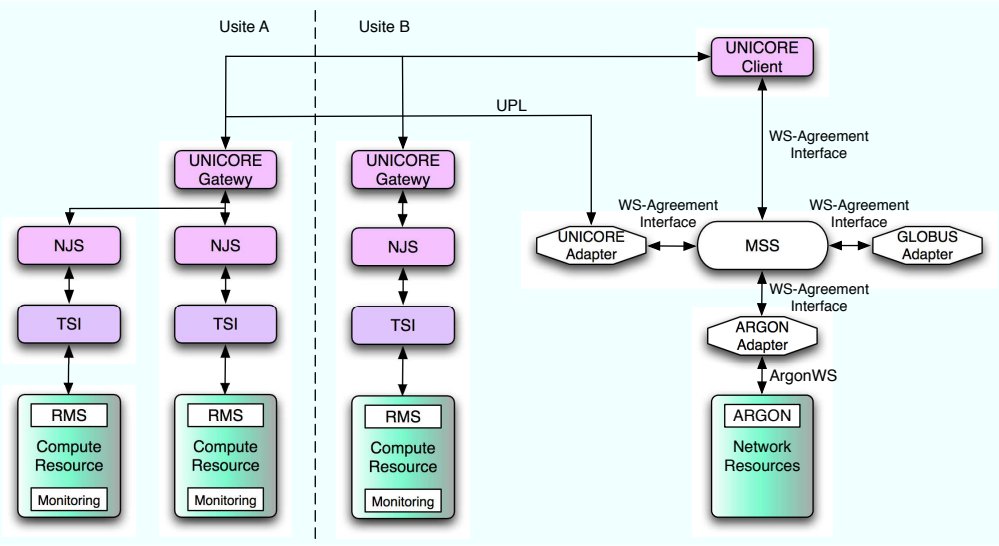

Figure 4: Architecture of the VIOLA MetaScheduling Service MSS.

In VIOLA a MetaScheduling Service was developed, which negotiates the time-slots with the different schedulers of the clusters and the NRMS and initiates the reservation of the nodes requested by the user. Figure 4 gives an

overview over the architecture showing the UNICORE systems network job supervisor (NJS) and the target system interface (TSI), which are responsible for handling the job transfer to the local compute resources. At the time agreed upon the resources will then be available for the users to run their distributed applications. Another use-case is a workflow spanning across several resources. The only difference to the use-case described before is the kind of temporal dependencies: While for the distributed parallel application the resources must be reserved for the same time, for the workflow use-case the resources are needed in a given sequence. Thus the orchestration service needs to negotiate the reservations such that one workflow component can be executed on the required resource after the preceding component is completed.

3.1.2 Grid Scheduler interoperation

As there is no single orchestrating service or Grid scheduler in a Grid spanning across countries and administrative domains we have to deal with multiple instances of independent Grid schedulers. Using resources from different domains requires co-ordination across multiple sites. There are two approaches either directly trying to negotiate with respective local scheduling systems or negotiation with the respective local orchestrator. The former solution requires local policies allowing a remote orchestrator to negotiate with local schedulers, which is in general not the case. In the second case there is one access point to the local resources, which then negotiates on behalf of the initiation orchestrator. Within its Task 1 - Definition of a Grid scheduling architecture - the CoreGRID Institute on Resource Management and Scheduling (RMS) is conducting research on the architecture and the protocols supporting Grid scheduler interaction. As a consequence of the work inside of CoreGRID members extended their work to the OGF and set-up a research group there. The Grid Scheduling Architecture Research Group (GSA-RG) has been initiated in 2004 and is co-chaired by CoreGRID RMS members since then. The engagement in the GSA-RG directly relates to Task 1 of the CoreGRID Institute on Resource Management and Scheduling.

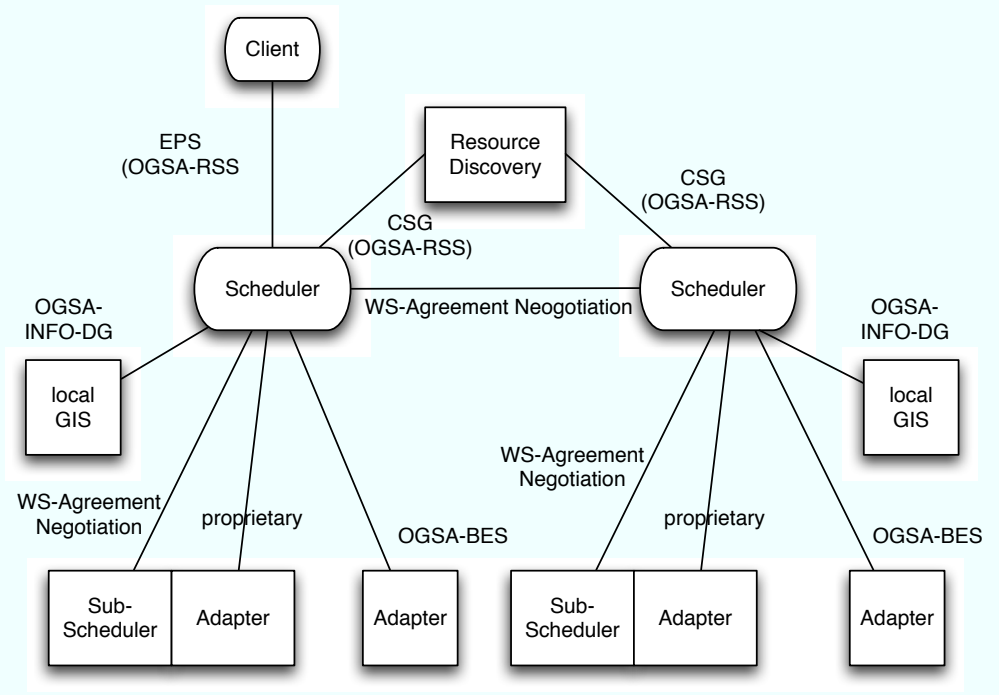

Figure 5: Grid Scheduling Architecture as discussed in the GSA-WG.

One of the outcomes of the joint work in both CoreGRID and the OGF was the preliminary definition of Grid Scheduling Architecture. Taking into account that the second approach described above also has a better scalability than the first one the OGF Grid Scheduling Architecture Research Group decided to consider this approach for the definition of Grid Scheduling Architecture. Figure 5 presents a high-level view on such a scheduling architecture.

For the communication between the different orchestration services or Grid schedulers, WS-Agreement as a language and a protocol to create SLAs was selected to achieve the necessary interoperability. Later at the end of the

negotiation process the resulting SLAs may be combined by the initiating resource orchestrator into one single agreement wrapping the agreements created with all individual service providers. This composed agreement is then the SLA between the orchestrator and his client, the end-user.

3.1.3 Business Grid Computing Project

Scenario
In the framework of the Japanese Business Grid Project [6] WS-Agreement was used in combination with the OGFs Job Submission Description Language (JSDL) [3] to express Service Description Terms for complex, wide-area jobs. The projects mission was to develop a business Grid middleware realising a next generation business application infrastructure. The focus was on combing resources from multiple service providers for a single complex wide-area job running business applications like a 3-tier Web application in a data centre. The main characteristics of the system are:

- Job submission is realised by a job description using WS-Agreement and JSDL, and the Application Contents Service (ACS).
- Brokering is used to allocate the necessary resources.
- Automatic deployment and configuration of programmes and data (including the necessary preparation of hosting environments) is realised by a language similar to the Configuration Description Language (CDL) [14], and the ACS.
- Management of heterogeneous resources is realised through Grid middleware agents providing a common interface (resource virtualisation).

WS-Agreement implementation and negotiation
We chose JSDL with extensions as the domain-specific language used within the Agreement. The link between JSDL constructs and WS-Agreement is realised through a convention specifying that the content of the JSDL element JobName corresponds to WS-Agreements ServiceName. We also extended the JSDL constructs in order to make the description of more complicated resource requirements easier. This resulted in two types of resource description information for each job: (i) Global resource information for describing the type of the job or providing general characteristics of the job (e.g. whether to allow automatic load control or not) and (ii) local resource information which describes the resource needed for each of the components that make up a pool to carry out the job. Several jobs, which are related - such as a 3-tier Web application for a Web shop and a batch job, which calculates the sales info every weekend - can be included in a single agreement.

The agreement template as described above is utilized in order to realise a wide-area business Grid. The aim here is to make it possible to share IT resources based on the contract/agreement among (i) distributed centres in an enterprise and (ii) trusted partners data centres (resource providers), thus making it possible for an Application Service Provider (ASP) to dispatch a complex job consisting of two 3-tierWeb applications spanning from a single portal over two sites (resource providers). The sequence of events, which occur in this scenario, is as follows (see also [15]):

1. The Application Service Providers manager prepares an Agreement Offer, which contains resource descriptions of both Job1 for one site and Job2 for another site, and sends the offer to the Global Grid Job Manager (GGJM).
2. The GGJM splits the Agreement Offer into two Agreement Offers, one for each sub-job, and offers the Agreement for Job1 to site A.
3. Site A decides to accept the Job, creates agreement1 and returns the Endpoint Reference (EPR) of the Agreement.
4. The GGJM stores the EPR of agreement1 internally and then offers the Agreement Offer for Job2 to site B.
5. Site B decides to reject the offer.
6. GGJM offers the Agreement Offer for Job2 to site C.
7. Site C decides to accept the offer, creates agreement2 and returns its EPR.
8. The GGJM stores the EPR of agreement2, creates an agreement for the complete job and returns it to the ASP manager.

3.1.4 Co-allocation - VIOLA MetaScheduling Service (MSS)

Scenario
In the VIOLA project [22] an optical testbed was implemented between multiple partners in Germany. Some of its main goals were the test of advanced network architectures, development of software for user-driven dynamical provision of bandwidth and test of advanced applications e.g. in the Grid area. A single instance of a MetaScheduling Service integrated into the UNICORE middleware is able to perform coordinated CPU and network bandwidth reservation between the clusters in the Grid, enabling distributed applications on these systems [23], [15].

WS-Agreement implementation and negotiation
The VIOLA MetaScheduling Service (MSS) [19] is responsible for negotiation of resource allocation with the local scheduling systems. It is implemented as a Web Service receiving a list of resources pre-selected by a resource selection service and returning reservations for some or all of these resources. The resource reservation is based on WS-Agreement. In the VIOLA project the MSS is utilized to co-allocate computational and network resources in a UNICORE-based Grid. Network resources are reserved through a WS-Agreement Interface with the ARGON Adapter. Computational resource reservations are negotiated through adapters with local scheduling systems also using WS-Agreement. Furthermore, the negotiation between the MSS and the UNICORE Client is based on WS-Agreement. This general architecture is illustrated in Figure 4. When a UNICORE Client wants to make a reservation, it sends the resource request to the MSS as a WS-Agreement template. The MetaScheduling Service then negotiates a potential start time for the Job and requests reservation of the network and computational resources. After successful completion of this reservation the MSS sends an End Point Reference (EPR) of the created WS-Agreement back to the UNICORE Client. It then generates a global MetaMPI configuration and transfers required data to the individual sites [5].

3.1.5 AssessGrid CCS

Scenario
AssessGrid [4] introduces risk management and assessment to Grid computing to facilitate a wider adoption of Grid technologies in business and society. Risk assessment helps providers to make decisions on suitable SLA offers by relating the risk of failure to penalty fees. Similarly, end-users get knowledge about the risk of an SLA violation by a resource provider that helps to make appropriate decisions regarding acceptable costs and penalty fees. A broker is the matchmaker between end-users and providers. The broker provides a time / cost / risk optimized assignment of SLA requests to SLA offers. AssessGrid introduces a framework supporting risk assessment and management for all three Grid actors (end-user, broker and resource provider). AssessGrid uses a distributed Resource Management System (RMS) called Computing Center Software (CCS or openCCS) [13]. CCS provides interfaces to UNICORE and Globus Toolkit.

WS-Agreement implementation and negotiation
In CCS a Grid resource broker provides transparent access to Grid resources by querying resource providers on behalf of end-users. For the end-user the broker acts as a pre-selector of Grid resource providers. The WS-Agreement protocol is currently being implemented for openCCS. The major negotiable SLA parameters in openCCS are:

- General parameters like number of nodes, amount of memory, job runtime
- Deadline for job completion
- Policies on security or migration
- Fault tolerance requirements

In the AssessGrid project the RMS is enhanced by risk assessment and risk management functionalities. A negotiator module is responsible for negotiating SLAs with external contractors using the WS-Agreement-Negotiation protocol. The negotiator defines the price, penalty and risk in the SLA according to policies. The risk is included in the SLA as an additional attribute. The Risk Assessor evaluates current monitoring information as well as aggregated statistical information to assess the risk for an SLA violation.

WS-Agreement is used in the communication of the user with the broker, the user with the provider and the broker with the provider. The user interface modifies the SLA template of the broker or provider based on the user prerequisites such as hardware architecture and available libraries and sends it back to the broker or provider in order to

receive SLA offers. Offers are presented to the end-user by the broker with price, risk of failure and penalty attributes. The end-user then either agrees or rejects such an offer [18].

3.1.6 AgentScape

Scenario
The AgentScape framework [1] provides mobile agents access to computing resources on heterogeneous systems across the Internet. AgentScape provides mobile agents coordinated resource usage across multiple domains. The negotiation of resource access for applications is based on WS-Agreements. An application can acquire time-limited contracts for resources. A mediator called domain coordinator (DC) in AgentScape represents multiple autonomous hosts and communicates with the mobile agent on behalf of these nodes. Agents can negotiate their options with DCs of multiple domains, being able to select the DC that provides the best offer.

WS-Agreement implementation and negotiation
The WS-Agreement based negotiation infrastructure of AgentScape allows agents to negotiate terms of conditions and quality of service of resource access with domain coordinators. Hosts providing resources are aggregated into virtual domains. The DC represents the hosts within a virtual domain in the negotiation process. The WS-Agreement interaction model is extended to allow a more sophisticated negotiation. In this extended negotiation model, hosts provide an agreement interface to their DC. The DC aggregates templates offered by hosts into composed templates and makes these available to agents. Agreement requests made by agents based on composed templates are received by the DC. The DC then negotiates an agreement with the hosts with the requested resources. The additional accept/reject interaction sequence allows agents to enter into negotiations with multiple providers and compare received offers. The extended negotiation architecture makes it also possible for a virtual provider to check an agents credentials before starting to negotiate with an agent [16].

The AgentScape negotiation architecture defines a set of resources that can be requested and used by agents as an AgentScape specific ontology. The following resources are included in this ontology: CPU time, communication bandwidth, amount of memory, disk space, web services that the agent is allowed to access and the number of calls of a web service that the agent is allowed to do.

After the negotiation phase, a host manager monitors and controls the resource usage to ensure that agreements are met. After the acceptance of an agreement, the agent has a limited time frame in which it has to migrate to the target location. If it fails to do so, the lease offer expires.

3.1.7 CATNETS

Scenario
CATNETS [18] simulates the application layer networks (ALN) environment by an economy, where the resources are for example processor time or storage space, while the economic actors are computers or web services. The application service and compute resource allocation of Application Layer Networks is broken down into two types of interrelated markets: A Grid resource market, where computational and data resources are traded and a service market where application services are traded. These services provide particular application functionality, e.g. query execution or molecule docking. In these separate markets complex services buy basic services, which buy raw resources. In this Catallaxy approach, the market is self-organizing which means that no centralized broker is required. In the prototype implementation the middleware is implemented as a set of simple, specialised agents using the light-weighted agents platform of the Decentralised Information Ecosystem Technologies (DIET) project. The agents provide for example access to markets, negotiations, object discovery and communication. The management of local resources is based on the WS-Resource Framework offered by Globus Toolkit 4. Middleware is further implemented using JXTA technology [21], [11].

WS-Agreement implementation and negotiation
WS-Agreement is used in the implementation of both the service market and the resource market. CATNETS defines separate bidding language for the service and the resource market, which are used by agents to submit bids for services or resources. These languages are mapped onto WS-Agreement via domain-specific schemas. The offers are encoded in XML using WS-Agreement and JSDL. In the resource market basic services can submit sell orders to the order books with WS-Agreement and the resource services can submit buy orders to the order books [10]. After submission

of all bids to the auctioneer, the allocation and the corresponding prices are determined, which results in an agreement. The activity on the service market is quite similar.

3.1.8 Job Submission Service

Scenario
Scenario The Job Submission Service (JSS) [12] is a broker aiming at identifying the set of resources that minimizes the Total Time to Delivery (TTD), or part thereof, for each individual job submission. In order to do this, the broker makes an a priori estimation of the whole or parts of the TTD for all resources of interest before making the selection. The TTD estimation includes performing benchmark-based execution time estimation, estimating file transfer times, and performing advance reservations of resources in order to obtain a guaranteed batch-queue waiting time. For resources not providing all information required or a reservation capability, less accurate estimations are performed. On the Grid resource, an authorization callout mechanism is used to validate that i) the requested reservation identifier exists and ii) the reservation actually was created by the same user that submits the job. JSS is currently used e.g. in NorduGrid and Swegrid.

WS-Agreement implementation and negotiation
The GT4/WSRF-based implementation of WS-Agreement is rather straightforward, with createAgreement requests forwarded by the AgreementFactory to a decisionMaker plugin, which interacts with the local resource management system. One such plugin is used to create advance reservations, and hence interacts with the batch system scheduler. JSS currently support the Maui scheduler, although others (supporting advance reservation) easily can be added. JSS also implement s a two-phase mechanism, where reservations will be released shortly after their creation, unless they are confirmed. This is implemented using WS-ResourceLifetime, as each Agreement is modelled as a WS-Resource. The terms used to negotiate advance reservations are

- number of CPUs,
- duration,
- earliest allowed start time,
- latest allowed start time,
- malleability flag.

The semantics of a malleable reservation is that the local scheduler may modify the reservation start time freely, as long as the starttime stays in the [earliest, latest] allowed start time window [7].

3.2 Requirements for a (re-)negotiation protocol

Initially the WS-Agreement specification also contained a more generic negotiation protocol. However, it was decided to remove this for version 1.0 since the negotiation approach included in the specification now already offered the necessary functionality for a large number of use-cases. Nevertheless, negotiation and re-negotiation of contracts in a distributed environment has been under discussion in Task 1 of the RMS and in the GRAAP group since the time of that decision. Meanwhile, the groups gathered a number of requirements that should be implemented by an extension of WS-Agreement for negotiation and re-negotiation of agreements.

The first approach towards negotiation was considering treating initial negotiation and re-negotiation as two separate issues: (i) Re-negotiation is done in the initial phase where the two parities negotiate on the Agreement terms and (ii) re-negotiation is needed after an Agreement has reached the Observed state. Recently we reached a now a common understanding, that the initial negotiation can be modelled as a re-negotiation of an agreement, which has not yet come into force. For that reason, we use the terms re-negotiation and negotiation in the rest of the document as synonyms.

There are two major features we think are necessary to make WS-Agreement an even more useful specification for SLAs:

- a malleable expiration time and

- a malleable list of resources

The first requirement is able to satisfy requests arising from dynamic business conditions, where one party may want to extend or shorten the time that an Agreement will be effective. The second requirement addresses situations - like the one described before in the Business Grid example - where resources will be allocated within a certain range specified in order to meet the SLAs. However, due to increasing resource demand there may be cases in which the Agreement Initiator would like to plan for enhancements to the current system. This could be done in the following two alternative ways: Re-negotiation of the original Agreement or creation of a new Agreement with references to the original Agreement.

Furthermore, the functionality and notion of Related Agreements was dropped at an earlier state of the specification. Related Agreements were intended to specify agreements with a dependency to other agreements, e.g. for agreements that rely on other agreements or for agreements that supersede a previous one as the result of a re-negotiation. Using End-Point-References (EPR) to the different Agreement instead makes the implementation easy as it allows borrowing from the agreement templates creation constraints.

The following list presents additional requirements for the re-negotiation:

- New agreements should have new EPRs,
- Re-negotiable agreements would also need to have new states in addition to the existing ones (depicted in Figure 12) to avoid race conditions.
- (Re-)negotiable terms should be tagged, depending on the initiating party (agreement initiator or agreement responder) either in the offer or in the template.
- Both service description terms (SDT) and guarantee terms (GT) should be re-negotiable
- Probably need new, more expressive faults to indicate why the agreement provider does not agree (in order to give hints to agreement initiator for what agreement provider can provide). Reasons could range from not compatible to the template, cannot meet this SDT/GT (with a pointer to the term), cannot meet this SDT by this value, etc. Also adding some syntactically meaningful limitations for SDTs the agreement responder might be able to fulfil in a rejection message could be helpful for the agreement initiator.
- Both agreement initiator and agreement responder should be allowed to ask for re-negotiation. However, this has impact on the protocol, as currently the agreement is maintained only on the side of the agreement responder.
- Probably need a supersedes/supersededBy attribute in the context
- Also, the capability of the agreement responder to return several alternative offers upon an SLA request might be helpful for the agreement initiator
- Offers need some kind of lifetime. Even if this is difficult in distributed systems, offers cannot be held open infinitely.
- In case of counter offers are acceptable responses the template should allow to restrict the counter offer.

The considerations and the requirements presented in this section lead to the proposals for re-negotiation protocols, which are described in the following sections.

3.3 Current proposals for a (re-)negotiation protocol

In this section we describe three approaches that have been discussed in the GRAAP-WG recently. Basically, they differ with respect to

- the primary focus (negotiation/re-negotiation)
- support for a 2-phase commit (2PC) protocol
- symmetry of the roles

3.3.1 A Contract Re-negotiation Protocol

The contract Re-negotiation protocol described an abstract, domain-independent protocol for the re-negotiation of contracts. Thus, the protocol is independent of the existing WS-Agreement specification. However, it is possible to use it together with WS-Agreement is some of the restrictions are modified from assumptions implicitly ruling the protocol to being implemented as domain specific policies that can be considered in the protocol depending on the domain. This protocol is based on the principles of contract law to make agreements with it legally compliant and allows for multi-round re-negotiation in an environment where messages may be lost, delayed and re-ordered. The protocol definition assumes an asymmetry of resource provision from resource consumption, which is reflected by putting a higher weight on the requirements of the service provider, e.g. with respect to avoiding potential loss due to the behaviour of a customer.

The protocol also allows the initiation of re-negotiation through non-binding enquiries to the other party so that an estimate as to how much it would cost to change the contract can be obtained before committing to a new contract. When re-negotiation is initiated the contract enters the re-negotiating state, which is a sub-state of contracted as the original contract is still in force, irrespective of the on-going re-negotiation. After successful re-negotiation current contract is in the superseded state as a new contract will have superseded this contract. The re-negotiation finite state machine is depicted in Figure 6.

The protocol includes several basic assumptions derived from the contract law

- offer messages are binding if accepted,
- all offers are acknowledged and,
- because of the risk of cheating by the customer, contract laws mailbox rule is invoked where a contract is formed when the accept message is sent by the offeree (i.e. the resource provider) and not when the accept is received by the offeror (i.e. the customer).

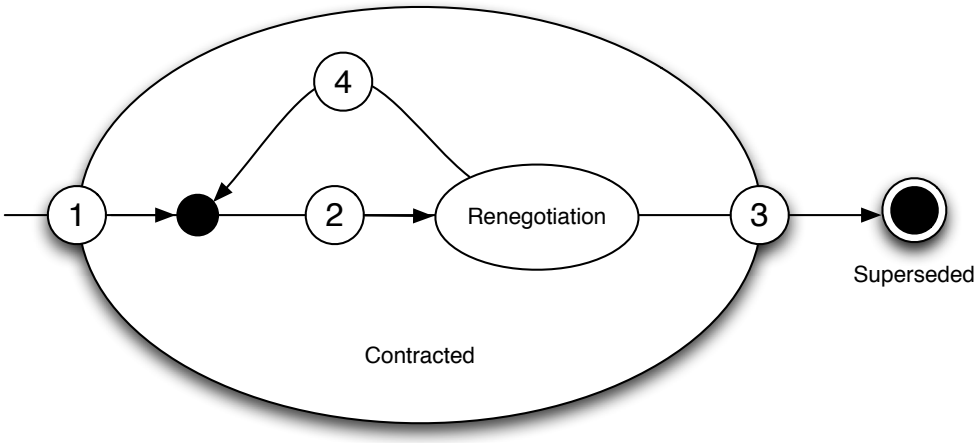

Figure 6: Re-negotiation state machine.

The proposed protocol offers the following messages:

- RenegotiationQuoteRequest sent only by the client asking for a (non-binding) quote for the re-negotiated contract
- RenegotiationQuote sent only by the provider delivering the quite
- RenegotiationOffer sent by the client as a binding request to form a new agreement
- RenegotiationOfferAck sent by the provider indicating that an offer has been received and is considered
- RenegotiationAccept sent by the provider indicating that the offer has been accepted and a new contract replacing the old one has been formed

- RenegotiationReject only sent by the provider indicating that the renegotiation offer was not accepted
- RenegotiationNotPossible can be sent by either party indicating that the re-negotiation is not/no longer possible

Several of the messages may be mapped directly to WS-Agreement messages, e.g. sending a RenegotiationReject is equivalent to a WS-Agreement fault message being sent, sending a RenegotiationAccept message is equivalent to WS-Agreements agreement responder role invoking the WSAG:AcceptAgreement operation.

3.3.2 Dynamic SLA negotiation based on WS-Agreement

Negotiation requires an iterative process between the parties involved. To rely on WS-Agreement and minimise the extensions to the proposed standard, this proposal suggests not to negotiate SLAs but to negotiate and refine the templates that can be used to create an SLA. Here, the focus is on the bilateral negotiation of agreement templates using a simple offer/counter offer model.

In order to use this model in the WS-Agreement protocol, a new function is introduced: negotiateTemplate. This function takes one template as input (offer), and returns zero or more templates (counter offer). The negotiation itself is an iterative process. In the following a simple negotiation process in three major stages is described. During the negotiation process the agreement initiator becomes the negotiation initiator. Accordingly, the agreement providers are negotiation responders. Figure 7 shows a sequence diagram for the negotiation of the agreement template.

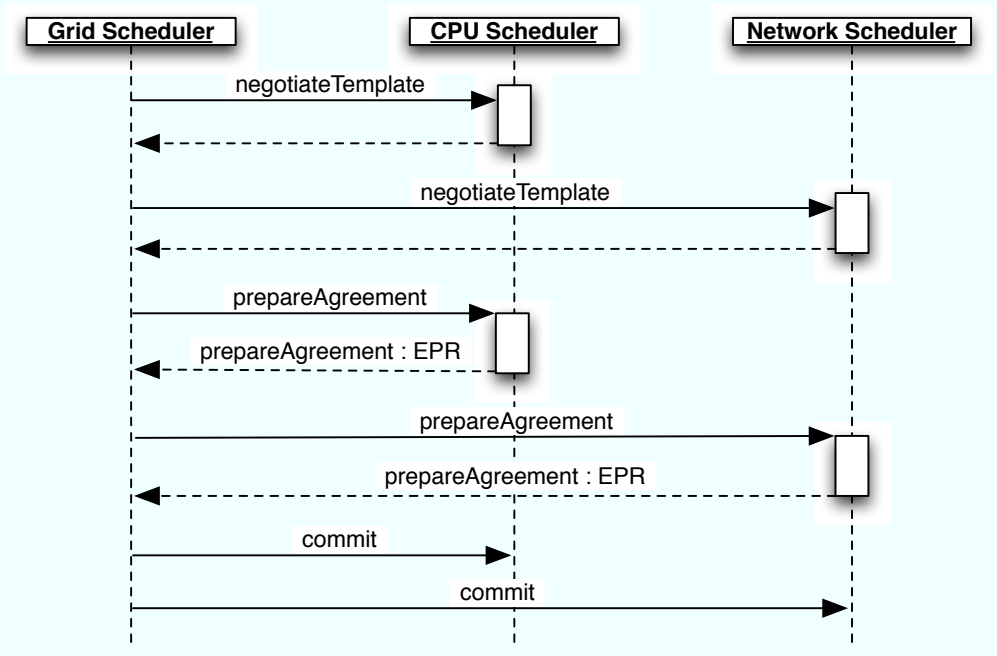

Figure 7: Extended WS-Agreement SLA negotiation.

1. Initialisation of the negotiation process
   First, the negotiation initiator initialises the process by querying a set of SLA templates from agreement providers by sending a standard WS-Agreement message, getResourceProperty request, to agreement providers (not shown in Figure 7). The initiator chooses the most suitable one as a starting point for the negotiation process. This template defines the context of the subsequent iterations. All subsequent offers must refer to this agreement template. This is required in order to enable an agreement provider to validate the creation constraints of the original template during the negotiation process, and therefore the validity of an offer.

2. Negotiation of the template
   After the negotiation initiator has chosen an agreement template, it will create a new agreement template based on the chosen one. The new created template must contain a reference to the originating template within its

context. Furthermore, the agreement initiator may adjust the content of the new created template, i.e. service description terms, the service property terms, and the guarantee terms. These changes must be done according to the creation constraint defined in the original template. Additionally, the negotiation initiator may also include creation constraints within the new created template. These constraints provide hints for the negotiation responder, within which limits the negotiation initiator is willing to create an agreement. After the initiator created the new agreement template according to its requirements, the template is send to responders via a negotiateTemplate message (as shown in Figure 7).
When the responder has received a negotiateTemplate message, it must first check the validity of the input document (refined template). This step includes (i) retrieving the original agreement template that was used to create the input document, (ii) validating the structure of the input document with respect to the originating template, and (iii) validating the changes of the content in the input document with respect to the creation constraints defined in the originating template. Once this is done, the agreement provider now checks whether the service defined in the request could be provided or not. If the service can be provided, it just returns the agreement template to the client, indicating that an offer based on that template will potentially be accepted. Otherwise, the provider employs some strategy to create reasonable counter offers. During this process the agreement provider should take into account the constraints of the negotiation initiator. Counter offers are basically a set of new agreement templates that base on the template received from the negotiation initiator. The relationship between dynamically created templates and original ones must be reflected by updating the context of the new templates accordingly. After creating the counter offers the provider sends them back to the negotiation initiator (negotiateTemplate response).

3. Post-processing of the templates
   After the negotiation initiator received the counter offers from the negotiation responder, it checks whether one or more meets its requirements. If there is no such template, the initiator can either stop the negotiation process, or start again from step 1. If there is an applicable template, the initiator validates whether there is need for an additional negotiation step or not. If yes, the initiator uses the selected template and proceeds with step 2, otherwise the selected template is used to create a new SLA.

For the final SLA creation the proposal suggests adding a new type of agreement that must be created in two phases: the first phase is a creation of the agreement triggered by a new prepareAgreement message resulting in a non-binding agreement (similar to the quote in the proposal 3.3.1. The second phase is initiated with a new non-standard Commit message as shown in Figure 7.

3.3.3 Protocol considerations based on the current WS-Agreement specification

If we consider supporting the symmetric approach of WS-Agreement where both sides can initiate an agreement there are basically two possibilities:

1. The party initially initiating the agreement (Agreement Initiator, AI) also initiates the re-negotiation (Modification Initiator, MI), and the original Agreement Responder (AR) is the party accepting or rejecting the Modification Offer (Modification Responder (MR).

2. The party having accepted the initial offer of the agreement initiator (Agreement Responder, AR) initiates the re-negotiation (Modification Initiator (MI) and the party initially initiating the agreement (Agreement Initiator, AI) accepts or rejects the Modification Offer (Modification Responder (MR).

The first possibility is depicted in Figure 8 showing a straightforward protocol approach. There might be multiple communication steps between the MI and MR until a Modification Offer is accepted by the MR and the EPR of the superseding Agreement is returned to the MI. This implements a 1-Phase Commit protocol (1PC).

The communication pattern of possibility 2 is slightly more complex (see Figure 9) since we need an additional message to deliver the EPR of the new Agreement to the MR after the MR has signalled that the offer will be accepted. This implements a 2PC and probably needs time-out mechanisms of the side of the MI to avoid waiting for a lost or never sent acceptance message of the MR.

In both cases the EPR of the existing agreement is sent together with the modification offer to allow the other party to verify that the offer is based on an existing Agreement and to check the offered modifications against the terms of the original Agreement.

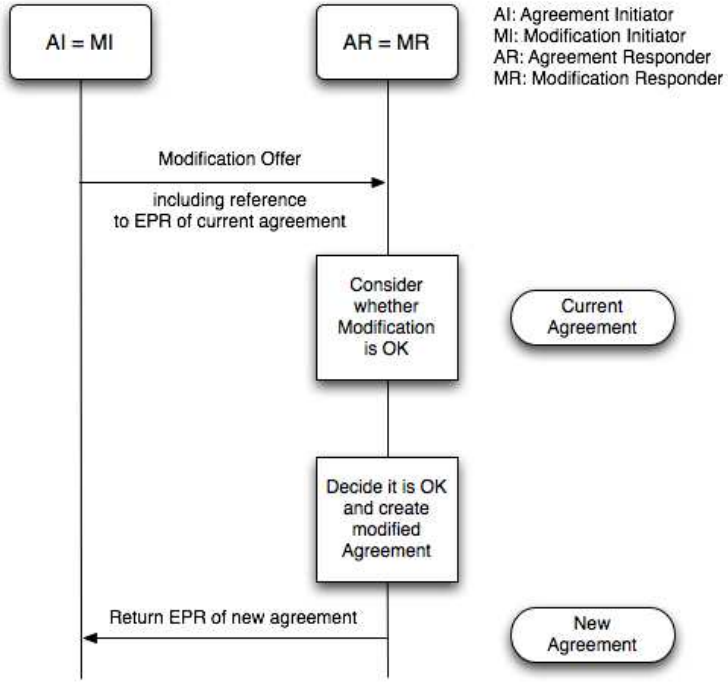

Figure 8: Agreement Initiator initiates Agreement re-negotiation.

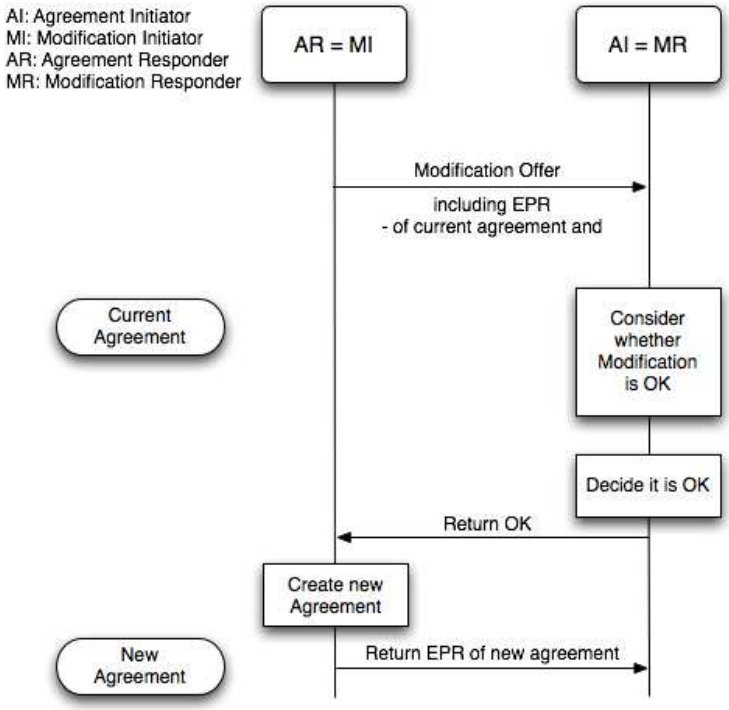

Figure 9: Agreement Responder initiates Agreement re-negotiation.

One possible rendering of the protocol avoiding the additional message of the MI to deliver the EPR of the new agreement to the MR is depicted in Figure 10. Here, the MI (who was the AR initially hosting the agreement factory) created a Pending Agreement and sends the EPR of the Pending Agreement together with the EPR of the current Agreement to the MR. Now, once the MR accepts the Modification Offer the MI changes the state of the Pending Agreement transforming it into an Agreement. The MR in turn now already has the EPR of the new Agreement.

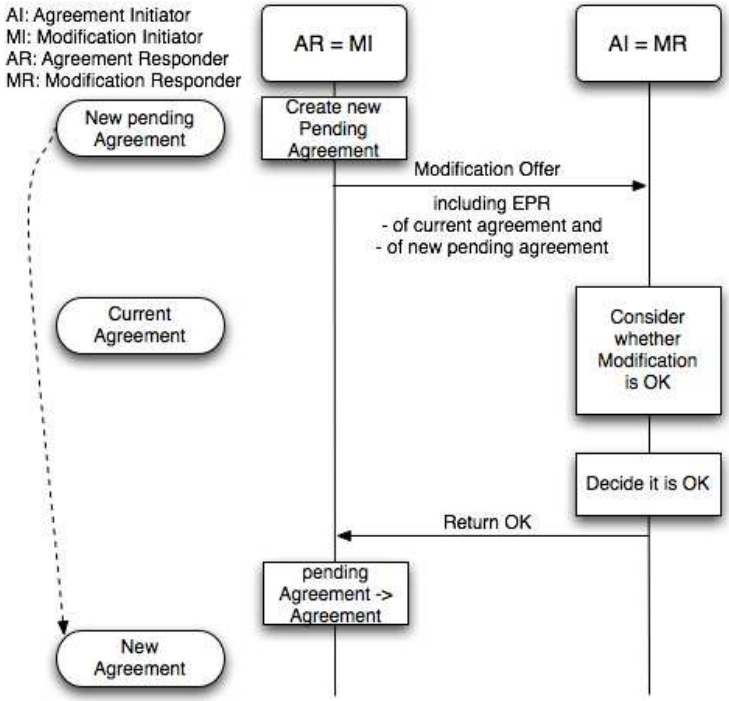

Figure 10: Agreement Responder initiates Agreement re-negotiation creating a pending agreement.

3.4 Towards a single negotiation protocol

Basically, all three approaches for a negotiation protocol work together with WS-Agreement requiring only minor changes to WS-Agreement. It is also common for the three approaches, to create a new Agreement as a result of a successful negotiation that is superseding the previous one instead of modifying the existing one. This also implies that the original agreement remains completely in force at least until the new superseding agreement has been created. The superseding agreement becomes effective once the negotiation has been completed successfully. In case the negotiation fails the original agreement remains in force.

Considering the differences as highlighted in the previous section, we suggest a separation of issues:

- Definition of the basic protocol, which will be common for all approaches
- Definition of policies, which may be used to guide the negotiation process

These policies that we expect to be domain dependent might be used, e.g. to

- restrict the possibility of initiation the negotiation to one of the parties (asymmetric approach),
- define the bindingness of the offers (for both cases AI=MR and AR=MR),
- define the time in the negotiation process when the superseding Agreement becomes effective

We suggest to implement WS-Agreement-Negotiation using the messages described in Section 3.3.1 describing the Contract Re-negotiation Protocol. As basic state machine for the negotiation process the one presented in Figure 6 will

be used. The exchange of modification offer messages and the respective acknowledgement messages (yes/no/EPR) will be based on refinements of the sequence diagrams (Figure 8, Figure 9, and Figure 10). Of course, there are additional issues that need further discussion with the GRAAP-WP, e.g.

- the agreement states of WS-Agreement might have to be extended to reflect an ongoing negotiation,
- the information that an agreement has been superseded by a new one after successful negotiation should available in the context together with the EPR of the superseding agreement
- the information that an agreement is superseding a previous one after successful negotiation should available in the context together with the EPR of the previous agreement

This report was used as a base for the discussion with the GRAAP-WG aiming to get a consensus on WS-Agreement-Negotiation during the OGF meeting in Barcelona, June 2008.

4 Monitoring of Agreements

Following the creation of an SLA with WS-Agreement both parties of the agreement need access to the state information of the agreement. In the case of WS-Agreement a set of resource properties is defined, that allows to monitor different aspects of the agreement. This information may be retrieved through the GetResourcProperty-function provided by the Web Service Resource Framework (WSRF), which is used by the WS-Agreement specification. Monitoring a SLA also empowers both parties to detect potential violations of the agreement and to take appropriate measures.

The WS-Agreement specification allows the monitoring of

- Agreement state
- Service Description Terms
- Guarantee Terms

The agreement itself may expose a set of properties that allows monitoring the agreement state. These properties are defined via the AgreementState port type.

- Port Type wsag:AgreementState
  The purpose of this port type is to define a resource document type for monitoring the state of the agreement.
- Resource Property wsag:AgreementState
  The property exposes an Agreement state for the whole agreement (see Figure 11)
- Resource Property wsag:ServiceTermState
  The property exposes a service state for each service description term that abstractly describes the state of a service (see Figure 12)
- Resource Property wsag:GuaranteeTermState
  This property represents a state of fulfilment for each guarantee term of the agreement (see Figure 13)

By default it is the task of the agreement responder hosting the agreement to provide the services that may gather the necessary data from the system environment to feed the information for allowing retrieving the different status information.

Data gathered from the system environment may include for example state of the scheduled job, its start time, or system monitoring data, e.g. from Ganglia. As said before, it is in the responsibility of the party hosting the agreement setting up the services to be able to extract the required information.

However, it might be desirable to delegate the task of gathering the information to a trusted third party in order to avoid that the agreement responder (which can be the service provider) also controls the source and the amount of data provided to determine whether the agreement is fulfilled or violated. To that end, WS-Agreement does not prescribe which party is providing this information on the various states.

Depending on the service description terms and the guarantee terms additional mechanisms might have to be in place to provide e.g. application or infrastructure specific data as for example transaction rates of a database or network throughput of a dedicated fibre connection.

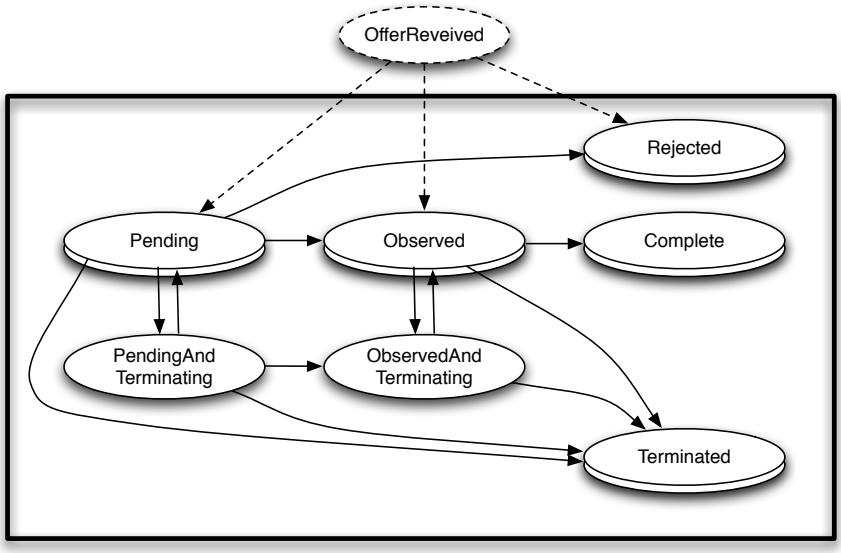

Figure 11: Agreement states exposed to an Initiator.

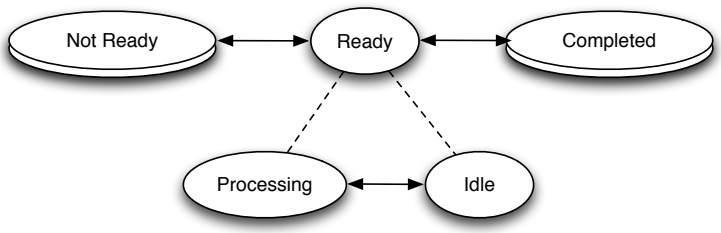

Figure 12: State model of the guarantee states.

An analysis of the mechanisms provided in WS-Agreement for monitoring of the various states shows, that the approach defined in the specification is (i) general enough and (ii) flexible enough to support the monitoring of agreements in their different states.

To that end, currently there is no need for additional mechanisms for the monitoring of SLAs based on WS-Agreement.

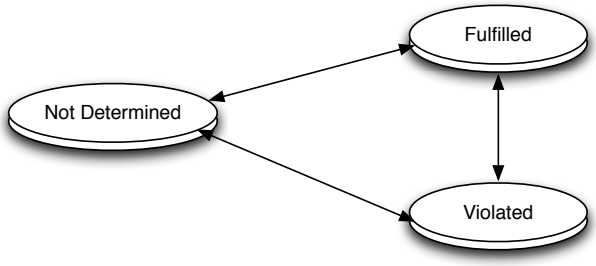

Figure 13: State model of the service term states.

5 Conclusion and Future Work

Members of the CoreGRID Institute on Resource Management and Scheduling (RMS) have a long record of activities in the Open Grid Forum (and its predecessors Grid Forum and Global Grid Forum). The GRAAP-WG has been initiated in 2002 and is co-chaired by CoreGRID RMS members since then. Even not yet being a full recommendation of the OGF the WS-Agreement specification has been implemented by a number of projects (ref CoreGRID TR). Negotiation and Monitoring of SLAs is of particular importance for commercial service providers. Standards are required to guarantee interoperability and the GRAAP-WG has become the focal point of these efforts, which is another reason for CoreGRID to heavily engage in this activity.

Based on the experience made with these implementations and following request from new projects for a more sophisticated protocol for negotiating and re-negotiating service level agreements the GRRAP-WG has started working on WS-Agreement-Negotiation after the WS-Agreement specification version 1.0 has become a proposed recommendation.

During OGF22 in Boston February 2008 the group has identified three approaches for a negotiation protocol each addressing different requirements of the parties involved in the negotiation. The group plans to consolidate these approaches into a single one, a first draft may be expected for OGF23 in June, a version ready to be submitted to the public comment process of the OGF is planned to be available at OGF24 in September.

In accordance with one of the overall objectives of CoreGRID, the integration of European Grid research, the focus of the work in the RMS is on integration and interoperability of the developments of the partners approaches and developments. Consolidating the ongoing R&D in the area of SLAs and their negotiation undertaken by several CoreGRID partners in several European countries is one facet of this integration effort. Moreover, there are also three CoreGRID fellowships working in this area; their results also leave its mark on the development of the negotiation protocol. The first is addressing use of SLAs for negotiation of resource usage in a High Performance Computing (HPC) environment [20]; the second one is investigating the use of SLAs for describing and managing software licenses, while the third ones subject is SLA & Contract Negotiation for the Grid, also focussing on industrial and commercial environments.

Finally, we will use this report to stimulate the discussion with the GRAAP-WG aiming to get a consensus on WS-Agreement-Negotiation before the next OGF meeting in Singapore, September 2008.

6 Acknowledgments

Most of the work has been carried out jointly within the CoreGRID Network of Excellence funded by the European Commission IST programme under grant #004265. Last not least the authors gracefully acknowledge the stimulating discussions in the GRAAP-WG and the contribution of GRAAP-WG members Karl Czajkowski, Univa UD,

Peer Hasselmeyer, European Laboratories, NEC, Bastian Koller, HLRS, Alexander Papaspyrou, Technical University Dortmund, Antoine Pichot, Alcatel.

References

[1] Agentscape Operating System. 18 July 2008 <http://www.agentscape.org>.

[2] A. Andrieux et al. Web Services Agreement Specification (WS-Agreement). Grid Forum Document GFD.107, Open Grid Forum, 2007.

[3] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification v1.0. Grid Forum Document GFD.56, Global Grid Forum, November 2005.

[4] AssessGrid - Advanced Risk Assessment and Management for Trustable Grids. Web site, 18 July 2008 <http://www.assessgrid.eu/>.

[5] Boris Bierbaum, Carsten Clauss, Thomas Eickermann, Lidia Kirtchakova;, Arnold Krechel, Stephan Springstubbe, Oliver Wäldrich, and Wolfgang Ziegler. Reliable orchestration of distributed mpi-applications in a unicore-based grid with metampich and metascheduling. In *Proceedings of the EuroPVM/MPI 2006*, volume 4192 of *LNCS*, pages 174 – 183, Bonn, Germany, September 2006. Springer.

[6] The Business Grid Project. Web siite 18 July 2008 <http://www.ipa.go.jp/english/softdev/sixth.html>.

[7] Erik Elmroth and Johan Tordsson. An interoperable, standards-based grid resource broker and job submission service. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 212–220, Washington, DC, USA, 2005. IEEE Computer Society.

[8] W. Sun et al. The Role of XML in Service Level Agreements Management. Network Management Research Center, Beijing Jiaotong University, IEEE, 2005.

[9] Grid Resource Allocation Agreement Protocol Working Group, 2008. Web site, 18 July 2008 <https://forge.gridforum.org/projects/graap-wg/>.

[10] L. Joita and O. Rana. WS-Agreement Use in CATNETS. Technical report, School of Computer Science and Welsh eScience Centre, Cardiff, UK, 2006.

[11] Liviu Joita, Omer F. Rana, Pablo Chacín, Isaac Chao, Felix Freitag, Leandro Navarro, and Oscar Ardaiz. Application deployment using catallactic grid middleware. In *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6, New York, NY, USA, 2005. ACM.

[12] Job Submission Service Project Homepage. 18 July 2008 <http://www.cs.umu.se/research/grid/brokering>.

[13] A. Keller. openCCS: Computing Center Software. Technical report, Paderborn Center for Parallel Computing, Paderborn, Germany, 2007.

[14] S. Loughran et al. Configuration Description, Deployment, and Lifecycle Management (CDDLM) Deployment API). Grid Forum Document GFD.69, Open Grid Forum, 2006.

[15] Heiko Ludwig, Toshiyuki Nakata, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. Reliable orchestration of resources using ws-agreement. In *Proceedings of the 2006 International Conference on High Performance Computing and Communications , (HPCC06)*, volume 4208 of *LNCS*, pages 753 – 762, Munich, September 2006. Springer.

[16] D.G.A. Mobach, B.J. Overeinder, and F.M.T Brazier. A WS-Agreement Based Resource Negotiation Framework for Mobile Agents. *Scalable Computing: Practice and Experience*, 7 (1):23 – 36, 2006.

[17] OGF – The Open Grid Forum. 18 July 2008 <http://www.ogf.org>.

[18] J. Padgett, I. Gourlay, and K. Djemame. AssessGrid System Architecture Specification and Developed Scenarios, V0.30. Technical report, AssessGrid Project, December 2006.

[19] Wieder; Philipp, Oliver Wäldrich, and Wolfgang Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In *Proceedings of the 6th International Conference, Parallel Processing and Applied Mathematics, PPAM 2005*,, volume 3911 of *LNCS*, pages 782 – 791, Poznan, Poland, September 2005. Springer.

[20] Hassan Rasheed, Ralf Gruber, Vincent Keller, Oliver WŁldrich, Wolfgang Ziegler, Philipp Wieder, Pierre Kuonen, Marie-Christine Sawley, Sergio Maffioletti, and Peter Kunszt. Ianos: An intelligent application oriented scheduling middleware for a hpc grid. Technical Report TR-0110, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, December 2007.

[21] Bjrn Schnizler, Dirk Neumann, Daniel Veit, Bjrn Schnizler, Dirk Neumann, Michael Reinicke, Werner Streitberger, and Torsten Eymann. WP 1: Theoretical and Computational Basis. Technical report, CATNETS Project, September 2005. 18 July 2008 <http://www.catnets.uni-bayreuth.de/fileadmin/publications/WP1-D1-Theoretical-and-Computational-Basis.pdf>.

[22] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN. Web site, 18 July 2008 <http://www.viola-testbed.de>.

[23] Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. Advanced techniques for scheduling, reservation, and access management for remote laboratories. In *Proceedings of the 2nd International Conference on e-Science and Grid Computing (e-Science 2006)*, Amsterdam, Netherlands, December 2006. IEEE.

[24] OASIS Web Services Resource Framework (WSRF) TC. 3 Apr 2006. Web site. Last access 18 July 2008: <http://www.oasis-open.org/committees/wsrf>.