An OGSA-based Quality of Service Framework

Rashid Al-Ali, [1,2] Kaizar Amin, [1,3] Gregor von Laszewski,[1,*]
Omer Rana [2] and David Walker[2]
[1] Argonne National Laboratory, U.S.A.
[2] Cardiff University, UK.
[3] University of North Texas, U.S.A.
[*] Corresponding Author: gregor@mcs.anl.gov

CONTENTS

Primary Contact:
Gregor von Laszewski
Argonne National Laboratory
9700 S. Cass Ave, Bldg. 221
Argonne, IL 60439, U.S.A.
phone: 630 252 0472
fax: 630 252 1997
email: gregor@mcs.anl.gov

An OGSA-based Quality of Service Framework

Rashid Al-Ali, [1,2] Kaizar Amin, [1,3] Gregor von Laszewski,[1,*]
Omer Rana [2] and David Walker[2]
[1] Argonne National Laboratory, U.S.A.
[2] Cardiff University, UK.
[3] University of North Texas, U.S.A.
[*] Corresponding Author: gregor@mcs.anl.gov

***Abstract*— Grid computing provides a robust paradigm to aggregate disparate resources in a secure and controlled environment. Grid architectures require an underpinning Quality of Service (QoS) support in order to manage complex computation- and data-intensive applications. However, QoS guarantees in the Grid context have not been given the importance they merit. In order to enhance the functionality offered by computational Grids, we overlay the Grid framework with an advanced QoS architecture, called *G-QoSM*. The G-QoSM framework provides a new service-oriented QoS management model that leverages from the Open Grid Service Architecture (OGSA) and has a number of interesting features: 1) Grid service discovery based on QoS attributes, 2) policy-based admission control for advance reservation support, and 3) Grid service execution with QoS constraints. This paper discusses the different components of the G-QoSM framework and presents an initial prototype implementation.**

I. Introduction

Grid computing [1], [2] has traditionally focused on large-scale sharing of distributed resources, sophisticated applications, and the achievement of high performance. The Grid architecture integrates diverse network environments with widely varying resource and security characteristics into virtual organizations (VO). Computational Grids offer a high end environment that can be exploited by advanced scientific and commercial applications.

Soft Quality of Service (QoS) assurances are made by Grid environments by the virtue of their establishment. Grid services are hosted on specialized "high-end" resources including expensive scientific instruments, clusters, and data storage systems. High connectivity is maintained between resources via dedicated high-speed networks. A well-established resource administration facilitates constant resource connectivity, resource monitoring, and fault tolerance. Hence, some preliminary level of QoS is provided by the committed members of the VO based on their pre-agreed Grid policy and their dedication in the overall collaboration. Nevertheless, the complexities involved in several critical Grid applications make it imperative to provide *hard* and *guaranteed* QoS assurances beyond that provided by the basic Grid infrastructure. Considering the increasing sophistication of Grid applications and new hardware under development [3] such provisions become an inherent requirement within the Grid architecture. This implies a need for a QoS management entity that facilitates a negotiation mechanism, where the clients can select the appropriate resources with QoS constraints that suit their needs.

Motivated by the need to overlay an advanced QoS framework on existing Grid architectures allowing them to support complex QoS requirements, we propose a QoS management framework, called as *G-QoSM*. Supporting the recent standardization efforts of the Global Grid Forum [4], the G-QoSM framework is based compatible with the latest Open Grid Services Architecture (OGSA) specification. The G-QoSM framework presented in this paper has a number of important features: 1) introduction of a 'QoS brokering service' as a Grid service, 2) introduction of a 'policy service' as a Grid

service and 3) introduction of a generic resource ‘reservation manager’, which has the following highlights:

- support for advance and immediate reservation,
- support for single and collective resource reservations (co-reservation),
- accommodation of arbitrary resource types, for example, compute, network and disk, and
- scalability and and flexibility through an object-oriented that is uses underlying resource characteristics at run-time.

The paper is structured as follows. In Section II we provide an overview of related research in the area of resource reservation to support QoS needs. In Section III-A we outline the general requirements of the Grid QoS model, and present the OGSA-based G-QoSM framework with reservation support. In Section VIII-A the reservation is defined, and we present a reservation admission control mechanism and reservation features. In Section IX implementation status is discussed, and prototype implementation screen shots are shown. A summary of future work concludes our paper.

II. Related Work

Immediate and advance reservation is considered in a wide variety of systems mostly in networking, communication, and distributed applications including distributed multi media applications (DMM). Hence it is of considerable interest to the Grid community.

- In the context of Grid computing, GARA [5] is a QoS framework that provides programmers a convenient access to end-to-end QoS. It provides advance reservations with uniform treatment to various types of resources such as network, compute, and disk. GARA’s reservation is a promise that the client/application who initiated the reservation will receive a specific level of service quality from the resource manager. GARA also provides reservation application program interface (API) to manipulate reservation requests, such as, *create*, *modify*, *bind* and *cancel*.
- NAFUR [6] describes the design and implementation of a QoS negotiation system with advance reservation support in the context of DMM applications. NAFUR aims to compute the QoS that can be supported at the time the service request is made, and at certain carefully-chosen, later times. For example, if the requested multimedia service with the desired QoS cannot be supported at the time the service request is made, the proposed approach allows the computation of the earliest time the user can start the multimedia service with the desired QoS.
- In [7] a resource broker (RB) model in the context of middleware for DMM application is proposed. The proposed RB has the following design goals: 1) advance and immediate reservation, 2) a new admission control scheme based on using a timely adaptive state tree (TAST) and 3) the RB processes brokerage requests for reservation, modifications, allocation and release. The new admission control based on TAST is used to make advance reservation decisions.
- In [8] advance reservation is formalized in the context of networking systems and the fundamental problem of admission control associated with resource reservation is introduced. Based on the authors literature review it is concluded that none of the previous approaches is sufficiently flexible to cover all potential needs of all users. The proposed solution to this fundamental problem is to separate the issue into a technical and a policy part supported by a specifying a generic reservation service description and a corresponding policy layer. This combination improves the flexibility of resource advance reservation compared to the other approaches.

None of the research efforts mentioned above address advance reservation in the context of service-oriented architecture, as in our approach. In general, resource reservation is not widely explored in service-oriented Grids. Nevertheless, the GGF Grid Resource Agreement and Allocation Protocol (GRAAP) Working Group, has produced a 'state of the art' document, which lays down properties for resource reservation in Grids [9]. We envision that our reservation model can be used to support the reservation properties outlined by the GRAAP-WG. The features that distinguish our work from existing QoS management approaches are that the

- generic QoS management service is not coupled to any specific resource type, or even limited to resource quantity;
- the object-oriented design and the abstraction approach gives the proposed service the ability to integrate with any brokerage system that supports web service interaction;
- dynamic information gathering and management, such as, resource characteristics and policy information improves scalability; and
- usage policy frameworks for resource providers/administrators and users to enable a fine-grained request specification.

In addition to the projects mentioned above, a general negotiation model called Service Negotiation and Acquisition Protocol (SNAP) is introduced in [10], which proposes a resource management model for negotiating resources in distributed systems. SNAP defines three types of SLAs that coordinate management across a desired resource set, and can, together, be used to describe a complex service requirement in a distributed system environment. Further,the resource interactions are mapped to well-defined platform-independent Service Level Agreements (SLAs) present the SNAP protocol to manage resources across different administrative domains via three types of SLAs: task SLA (TSLA), resource SLA (RSLA) and bind SLA (BSLA). The TSLA describes the task and the RSLA describes the resources needed to accomplish the task in the TSLA. The BSLA associates the resources from the RSLA and the application 'task' in the TSLA. The SNAP protocol necessitates the existence of resource management entity that can provide promises on resource capability; for example, RSLA. Therefore, our reservation model can encapsulate such a requirement and implement the RSLA negotiation.

III. The Proposed QoS Framework

In this section we introduce the proposed Grid QoS Management framework. We outline general requirements for the framework, and then we provide discussion on QoS management and the proposed system.

A. Requirements

The proposed framework must adhere to certain important requirements:

a) Service Discovery: The system should be able to discover services based on QoS attributes. These attributes are a) quantitative and b) qualitative. For example, quantitative attributes include computation, networking and storage requirements, while qualitative attributes include the degree of service reputation and service licensing cost. To support service discovery based on these attributes, a discovery mechanism needs to be employed within the proposed framework.

b) Resource Advance Reservation: The system should support mechanisms for advance, immediate, or 'on demand' resource reservation. Advance reservation is particularly important when dealing with scarce resources, as is often the case with high performance and high end scientific applications in Grids.

c) Reservation Policy: The system should support a mechanism which facilitates Grid resource owners enforcing their policies governing when, how, and who can use their

resource, while decoupling reservation and policy entities, in order to improve reservation flexibility. [8].

d) Agreement Protocol: The system should assure the clients of their advance reservation status, and the resource quality they expect during the service session. Such assurance can be contained in an agreement protocol, such as Service Level Agreements (SLAs).

e) Security: The system should prevent malicious users penetrating, or altering the data repositories that holds information about reservations, policies and agreement protocols. A proper security infrastructure is required, such as Public Key Infrastructure (PKI).

f) Simple: The system should have a simple design that requires minimal overheads in terms of computation, infrastructure, storage, and message complexity.

g) Scalability: The system should be scalable to large numbers of entities, as the Grid is a global scale infrastructure.

B. Grid Quality of Service Management

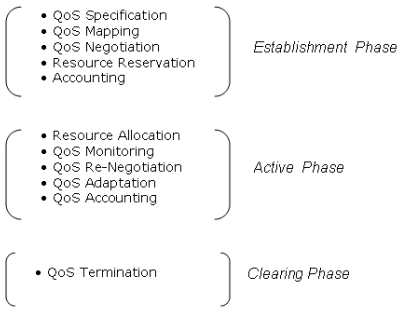

Fig. 1. QoS Management Phases.

Grid Quality of Service Management (G-QoSM) is a new approach to supporting Quality of Service (QoS) management in computational Grids, in the context of Open Grid Service Architecture (OGSA). QoS management includes a range of activities, from resource selection, allocation, and resource release; activities applied in the course of a QoS session. A QoS session includes three main phases: i) the establishment phase, ii) the active phase, and iii) the clearing phase [11]. These phases include a number of QoS functions as depicted in Figure 1. In QoS-oriented architectures, during the 'establishment phase', a client's application states the desired service and QoS specification. The QoS broker then undertakes a service discovery, based on the specified QoS properties, and negotiates an agreement offer for the client's application. During the 'active phase', additional activities, including QoS monitoring, adaptation, accounting and possibly re-negotiation, may take place. The 'clearing phase' is responsible to terminate QoS session, either through resource reservation expiration, agreement violation or service completion, and resources are freed for use by other clients.

Quality of service management has been explored in a number of contexts, particularly for computer networks [12], multimedia applications [13] and Grid computing [5]. Regardless of the context, a QoS management system should address the following needs:

- Specifying QoS requirements.
- Mapping QoS requirements to resource capabilities.
- Negotiating QoS with resource owners - where a requirement cannot be exactly met.
- Establishing service level agreements (SLAs) with clients.
- Reserving and allocating resources.
- Monitoring parameters associated with a QoS session.
- Adapting to varying resource quality characteristics.
- Terminating QoS sessions.

The Grid QoS Management (G-QoSM) [14] framework aims to operate in service-oriented architectures. It provides three main functions: 1) support for resource and service discovery based on QoS properties, 2) support for providing QoS guarantees at middleware and network levels, and establishing

Service Level Agreements (SLAs) to enforce these guarantees, and 3) providing QoS adaptation for the allocated resources. The G-QoSM delivers three types of QoS levels: *Guaranteed*, *Controlled Load* and *Best Effort* QoS. At the 'guaranteed level', constraints, related to the QoS parameters of the client, need to exactly match the service provision. 'Controlled load' is similar to the 'guaranteed' level, with the exception that less stringent parameter constraints are defined, and the notion of range-based QoS attributes is used along with range-based SLAs. At the 'best effort' QoS level the resource manager has full control in choosing the QoS level without constraints, corresponding to the default case when no QoS requirements are specified.

The Grid QoS Management (G-QoSM) [14] framework aims to operate in service-oriented architectures, and to provide three main functions: 1) support for resource and service discovery based on QoS properties, 2) support for providing QoS guarantees at middleware and network levels, and establishing Service Level Agreements (SLAs) to enforce these guarantees, and 3) providing QoS adaptation for the allocated resources. The G-QoSM delivers three types of QoS levels: Guaranteed, Controlled Load and Best Effort QoS. At the 'guaranteed level', constraints, related to the QoS parameters of the client, need to exactly match the service provision. 'Controlled load' is similar to the 'guaranteed' level, with the exception that less stringent parameter constraints are defined, and the notion of range-based QoS attributes is used along with range-based SLAs. At the 'best effort' QoS level the resource manager has full control in choosing the QoS level without constraints, corresponding to the default case when no QoS requirements are specified.

The G-QoSM is an ongoing project, previously investigated and implemented in the context of Globus toolkit (GT) 2.0, [14] [15] using GARA framework to provide QoS support for 'compute' resources. However, with the emergence of Service-Oriented Grids, and Open Grid Service Architecture (OGSA) [16] it is necessary to introduce new features to the G-QoSM, to make it OGSA-enabled and GT3 compliant. In this new G-QoSM architecture GARA is not utilized, and is replaced by a new reservation manager, policy service, allocation manager and a newly-developed Java API for a Dynamic Soft Real Time (DSRT) scheduler [17]. The new features in the OGSA-enabled G-QoSM are:

- Introduction of a 'QoS brokering service' as a Grid service.
- Introduction of a new generic resource 'reservation manager'.
- Introduction of a new 'policy service' as a Grid service.
- Introduction of a framework that is OGSA-enabled and can be instantiated in the context of GT3.

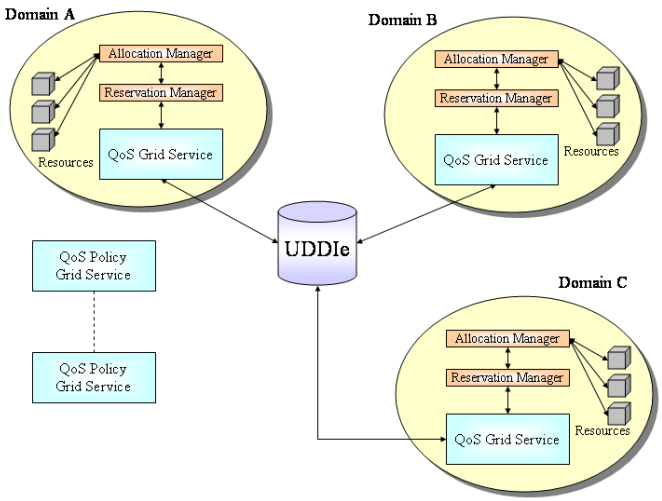

Fig. 2. Framework Architecture.

Figure 2 shows the new G-QoSM OGSA-enabled architecture. The main components are: QoS Grid service, an extended version of the Universal Description Discovery and Integration (UDDIe), resource reservation manager, resource allocation manager and policy Grid service.

IV. QoS Grid Service

QoS Grid Service (QGS) is the focal point of this architecture and exists in every domain, with a domain characterized

by an IP subnet or Globus site. The QGS interacts with the client's application, the QoS selection Service, the reservation manager, and the policy Grid service to support:

h) Interaction with Client's Application: To primarily capture the service request with QoS constraints, and to negotiate a QoS agreement SLA interaction with client's application is needed. This negotiation can be summarized as attempting to find the 'best match' service, based on given properties and priority levels, for example, one might request that cost has a higher priority than service reliability, and the matching process should comply with such a requirement. Once the best service match is found, and corresponding resources are reserved, an agreement offer is proposed to the client's application. If the proposed agreement is approved, it becomes a commitment, and the QGS regards this agreement as a fixed guarantee. Otherwise resources are released and no agreement takes place.

i) Interaction with the QoS Selection Service: To support basic concept queries a QoS selection service is provided with QoS constraints similar to the one supplied by the client's application. It's main function is to provide information and the decision for selecting the best service. Normally, the selection service replies with a list of service matches, which necessitates the QGS selecting one of the returned services. To make the best selection, the client's application should associate an importance level value to each required QoS attribute in the initial service request. We adapted a selection algorithm based on a Weighted Average (WA) concept, taking into account the proportional value of each QoS attribute, using the importance level supplied by the user in the 'service request', rather than each attribute being treated equally. The 'importance level' associates a level of importance or priority, such as High (H), Medium (M) and Low (L), to each QoS attribute, with this importance level mapped to a numerical value (real number). The algorithm computes the WA for every returned service and selects the service with the highest WA.

j) Interaction with Reservation Manager: After selecting a Grid service the functional requirements, required in support of the reservation, are extracted and formulated as resource specifications. These resource specifications are then submitted to the reservation manager for resource reservation, and a reservation 'handle' is returned in the case of a successful reservation. This reservation handle can be later used to claim, or manipulate, the reservation.

k) Interaction with Policy Grid Service: Interaction with the policy grid service enables the QGS to capture policy information necessary to validate the service request. For example, to discover if there is any limitation on resource utilization per service, or the class of service requested. The QGS validates the service request by applying the rules obtained from the Policy Grid Service.

V. QoS Service Registry

UDDIe [18] forms the service registry system in the context of the G-QoSM framework – Figure 2 shows where the UDDIe integrates with the rest of the framework components. It is used primarily as a registry to publish QoS attributes of services, and subsequently to search for services based on QoS attributes. The publication process, as per the Open Grid Service Architecture (OGSA) specifications, requires that service providers supply two separate WSDL documents, namely, service interface and implementation documents that describe the syntax and semantics for accessing the service. Current OGSA specification does not specify a QoS structure to be included in the services' WSDL documents. Every service in our framework has two interfaces: functional and management interfaces. The functional interface describes how the given service could be accessed, whereas the management interface describes attributes related to QoS and performance characteristics associated with the service. We incorporate the

QoS properties, such as the resources needed to execute the Grid service and service cost, etc., in the service management interface. With this extension to the WSDL document, the service provider would be able to describe their services in terms of QoS properties. In order for this extension to be recognized by a registry system such as UDDI and hence be searched, the UDDI registry needs to be extended to recognize these additional attributes. We design and implement 3 search capabilities by extending UDDI: service properties, service leasing and range-based search. These properties together can be used to search for stored services based on their properties – rather than their keys or `TModels` – as undertaken in a standard UDDI implementation. Documents stored within the registry also have a lease associated with them, and an event manager to support these leases has been implemented in UDDIe. In this way, the authors are able to search for documents which match a given range of QoS attributes for supporting various classes of QoS. More detailed discussion on the UDDIe enhancement can be found in [18].

VI. QoS Allocation Manager

The Allocation Manager's primary role is to interact with underlying resource managers for resource allocation and de-allocation, and to inquire about the status of the resources. It has interfaces with various resource managers employed in this framework, namely, the Dynamic Soft Real Time Scheduler (DSRT) [17] and a Network Resource Manager (NRM). It associates the execution of Grid services with a previously-negotiated SLA agreement, which process, of associating Grid services with SLAs, is beyond the scope of this paper. The Allocation Manager further interacts with adaptive services to enforce adaptation strategies, with more details on adaptation to be found in [15].

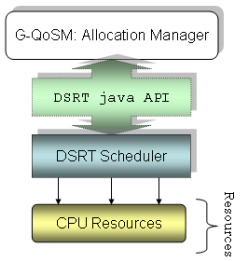

Fig. 3. G-QoSM and DSRT Integration.

A. DSRT Resource Manager

The DSRT [17] is a user-level soft real-time scheduler, based on the changing priority mechanism supported by Unix and Linux operating systems. The highest fixed priority is reserved for the DSRT and the real-time process admitted by the DSRT can then run under the DSRT scheduling mechanism. The real-time process can thus be scheduled to utilize a specific CPU percentage. Therefore, the compute QoS supported by the DSRT can be specified in terms of CPU percentage; for example, a real-time process might request the allocation of 40% of the CPU. Figure 3 shows the DSRT resource manager integrated with the new G-QoSM.

B. Network Resource Manager

The Network Resource Manager (NRM) is conceptually a Differentiated Services (Diffserv) Bandwidth Broker (BB) (a concept described in [19]), and manages network QoS parameters within a given domain, based on agreed SLAs. The NRM is also responsible for managing inter-domain communication, with NRMs in neighboring domains, to coordinate SLAs across domain boundaries. The NRM may communicate with local monitoring tools to determine the state of the network and its current configuration. Figure 4 depicts a NRM managed Diffserv domain.

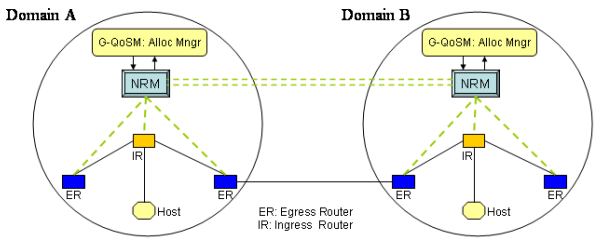

Fig. 4. G-QoSM and NRM Integration.

VII. QoS Policy Service

Policy Service is a Grid service aiming to provide dynamic information about the domain-specific resources' characteristics and the domain's policy concerning when, what and who is authorized to use resources. This policy service relies heavily on the existence of a policy repository, such as, the 'policy controller' in our framework. Resource owners include in the policy repository domain-specific rules; for example, resource capacity allowed to be utilized with user authentication, time of the day and class of service. These rules are utilized by the policy service manager to provide information on resource characteristics and domain policies. Having a separate policy manager as a Grid service allows the following advantages:

- The ability for resource owners to update their policy repository without interfering with other broker services.
- The resource owner may delegate a remote 'super' policy service to act as the policy controller of their resources. Similarly, a policy service might control more than a single administrative domain.
- Decoupling the policy service from other broker services, allows the ability to dynamically change resource usage policy and system scalability.

VIII. QoS Reservation Manager

Reservation support plays a major role in QoS-oriented architecture. In a shared resource environment, such as Grids, QoS brokers can provide promises on delivering certain resource quality to their clients, if, and only if, a reservation mechanism exists. A reservation can be viewed as a promise from the resource broker to clients on expected quality. Advance resource reservation is defined as: *a possibly limited or restricted delegation of a particular resource capability over a defined time interval, obtained by the requester from the resource owner through a negotiation process* [9]. As pointed out earlier, resource reservation can be categorized into: (a) Advance reservation and (b) Immediate or 'on demand' reservation, and can be for a specified duration, or indefinite. In the proposed reservation manager, we support advance/immediate reservation for a specified duration. Indefinite reservation is undesirable as it introduces blockages, which may result in a waste of unused resources. An important feature of this reservation approach is support for the co-reservation of various resources in service Grids.

In this section we further discuss the formal definition of reservation, admission control and outline reservation features.

A. Reservation Definition

We define a reservation model for collective Grid resources, with as few restrictions as possible, to increase the flexibility of the admission control. The fundamental problem with advance reservation, as discussed in literature [8], is that when an advance reservation is granted, the time from when the reservation is submitted until the start time, is called 'hold-back time', and to utilize, or grant, reservations during hold-back time is a complex problem. The problem arises when clients request immediate reservation for an indefinite period, which may, obviously, overrun a previously-granted advance reservation. A number of solutions are proposed to solve this

problem; for example, all reservations, including immediate reservation, must be specified within a time frame (i.e. indefinite reservation is not supported); another solution proposes to partition resources for immediate reservation, and advance reservation with specified durations. In this model we opt for the first proposal; that all reservations must be accompanied by duration specifications. We consider this a valid assumption as we deal with high performance resources, and application domains, like scientific experiments or simulations, means there is prior knowledge of the need for such resources, and no ad-hoc requests for simple resources.

We formally define reservation R in terms of the following (5) parameters:

- t_s : reservation start time
- t_e : reservation end time
- cl : reservation class of service
- r_i : each resource i has a resource type. Such types can be "compute", "network", and "disk", ...
- $c(r,t)$: is a function that returns the capacity of resource r at time t.

With these notation one can express reservation request $R(t_s, t_e, cl, \{(r_1, c(r_1)), .., (r_n, c(r_n))\})$ as a co-reservation for n resources, with start time t_s and end time t_e, using QoS reservation class cl on r_i with the associated capacities $c(r_i)$. We also introduce in this definition the concept of pre-emption priority, which has been explored in the context of networking and communication service [8]. The pre-emption priority is that when the reservation is not in effect, either before or after the reservation period, the job, or service that makes use of the reserved resource is not turned down or eliminated, but is rather assigned a low priority value, which means switching its status from 'guaranteed' to a 'best effort' type of service. In practice to support this concept the underlying resource manager should be a priority-based system, such as the Dynamic Soft Real Time (DSRT) scheduler [17]. This feature is very useful in protecting applications when reservations expired.

B. Admission Control

Admission control is the process of granting/denying reservation requests based on a number of factors, such as, the actual load of the specified resource, the policy that governs who, how and when reservation for resource usage should be granted. To perform an admission control process an admission control mechanism must be employed. We formally describe our admission control mechanism as a 'Boolean' function that returns $true$ or $false$ for a reservation request R at time t. $true$ means the reservation can be granted for the given time t with the resource specifications, and $false$ means otherwise. To further define the admission control function algorithm, we first define the notion of resource load L at time t:

$$L(r_j, t) = \sum_{i=1}^{g(t)} c(r_j, t)$$

where $g(t)$ is the number of granted reservations for time t and $c(r_j, t)$ is the amount of capacity reserved on the resource type j at time t.

We also need to define resource total capacity as the maximum capacity the underlying resource can provide; formally $max(r_i)$ is the maximum capacity that the resource i can provide.

With the above basic primitives, we can now define the algorithm for the admission control function.

Algorithm 1. Admission Control Function

```
1  Input: reservation R(t_s, t_e, cl, {(r_1, c(r_1)), .. , (r_n, c(r_n))})
2  Output: boolean
3  for i = 1 to n
4    for t = t_s to t_e
5      if c(r_i, t) > (max(r_i) − L(r_i, t)) then
6        return false
7      end if
8    end for
9  end for
10 return true
```

C. Reservation Features

As the reservation manager presented in this work operates in a Open Grid Service Infrastructure (OGSI), the service has a number of 'operations' can be used by other components. These operations are implemented as an API with a set of primitives, briefly described as:

- *reserve:* is invoked by sending a reservation tuple R, which will reply with a 'reject reservation', if the reservation cannot be granted. Otherwise it returns a reservation 'handle', a reference for the newly-made reservation request.
- *isAvailable:* where one might be interested in checking the status of some resource prior to placing the actual reservation; this operation supports such a request and returns a Boolean result accordingly.
- *nextAvailable:* is an interesting operation as it can be used for the purpose of 'counter-proposals'. A brokering service can use this if the user's request for reservation cannot be granted, rather than replying with a Yes/No type answer as is the case with most reservation systems, the operation can reply with a 'No' answer and a counter-proposal for the next availability.
- *extend:* can modify a previously-made reservation by extending it for a specified duration.
- *find:* finds a previously-made reservation, and replies with all details about the reservation.
- *cancel:* cancels a previously-made reservation.

With this set of reservation operations on the reservation manager a higher level brokering service, or agent, can make use of this manager to provide immediate reservations, and reservations in advance, and also manipulate these reservations.

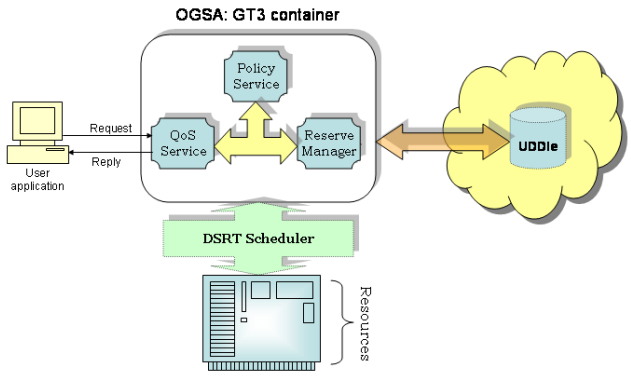

Fig. 5. Current Implementation Architecture.

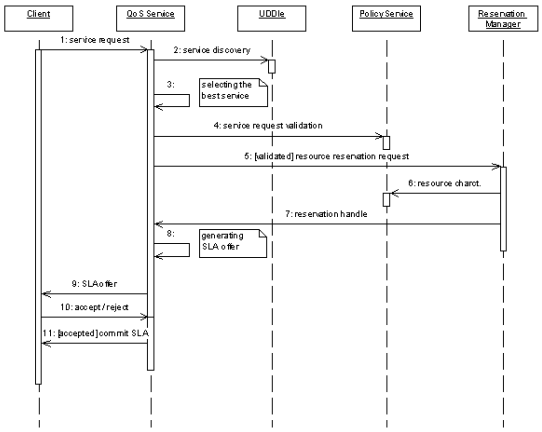

Fig. 6. A UML Sequence Diagram for the QoS Agreement Negotiation.

IX. Implementation Scenario

The implementation test-bed is based on the following technologies: Globus Toolkit 3.0, Linux Red-Hat 9.0, JSDK 1.4.2, Dynamic Soft Real Time (DSRT) schedule, Java CoG Kit 1.0 [20], OGCE component tools, UDDIe and Tomcat 4.1 application server. Figure 5 shows the implementation architecture.

In this scenario a user application uses the Java CoG Kit tools to invoke the QoS Grid Service (QGS) to negotiate a Service Level Agreement (SLA) to execute the desired Grid service. The SLA is generated as a result of a negotiation process takes place during the 'Establishment phase'. Further,

Fig. 7. A User interface screenshot, showing the user entered a 'service_request' and the QGS rplied with a service SLA offer.

a number of components are involved in the negotiation, Figure 6 is a UML interaction sequence diagram shows the interacting between the various components. Figure 7 is a screen shot of the user application GUI with the required parameters, in the upper half of the screenshot, to start the negotiation. The service request contains QoS constraints, stated by the user application, such as the maximum budget, the lowest service reputation, networking requirements, and the associated importance level or priority level for each QoS attribute. Once the request is submitted to the QGS, the QGS carries out the discovery process by contacting the UDDIe servlet for similar services with the QoS specified. The QGS then implements a selection algorithm to select the best match based on the user's quality preference (importance/priority) stated in the service request. After the selection process, the QGS contacts the policy service to validate the request. When the request is validated, a reservation manager is contacted to co-reserve the required resources, with the reservation manager contacting the policy service to obtain resource configurations and reservation policies. Having selected a service and reserved the required resources, the QGS proposes a SLA offer for the user application and expects an agree/reject reply within a pre-defined time frame. Figure 7, the lower half of the screenshot, shows the proposed SLA offer. If the user application accepts the offer then the SLA is stored in the SLA repository and becomes a commitment, and subsequently the user application can claim this Grid service with the reservation parameters, stated in the SLA, by contacting an execution service with the given SLA-ID, which execution service is beyond the scope of this paper. Otherwise, if the user application rejects the offer, or time elapses without reply from the user, the SLA offer and the associated resource reservation is canceled. The user application can repeat this process a number of times, with altering QoS parameters to demonstrate negotiation process.

X. Conclusion and Future Work

In this paper, we propose a QoS service model in service-oriented Grids comprising a brokering service and a number of supporting modules, including policy service, reservation manager, allocation manager, and QoS-aware UDDIe. Throughout this paper we describe the individual components of our framework and outline their patterns of interaction. We also discuss an OGSA compliant prototype implementation for our G-QoSM architecture.

The important features of our approach are: the QoS manager is a Grid service and dynamically interacts with a reservation and policy service modules, which makes it possible for resource owners to update/modify their policies during runtime; and the reservation is abstracted as a generic service for co-reservation support, which makes it very suitable for distributed computing, such as Grids. This abstraction allows the reservation service to operate with any underlying resources,

without previous knowledge of the resource characteristics, with the association of resource characteristics taking place during run-time by querying the policy service. This novel feature demonstrates scalability - highly desirable in Grid infrastructure.

As a future topic, we plan to investigate in new approaches for 'Grid Execution service' that is binding to Grid services based on previously negotiated QoS agreements SLAs. Investigate further in applying QoS techniques, during QoS active session, to enforce the agreed upon SLAs elements. These QoS techniques comprising QoS monitoring, re-negotiation and adaptation. It is also planned to integrate this framework with a particular scientific application domain, namely, using Grid computing for analysis of nanoscale structures, utilizing the newly developed experiment technique called position-resolved diffraction, as part of Argonne National Laboratory's advanced analytical electron microscope [21].

ACKNOWLEDGMENT

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus Project research and development. The Java CoG Kit Project is supported by DOE SciDAC and NSF Alliance.

REFERENCES

[1] G. von Laszewski, G. Pieper, and P. Wagstrom, "Gestalt of the Grid," in *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, ser. Series on Parallel and Distributed Computing. Wiley, 2003, (to be published). [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--gestalt.pdf

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, vol. 15, no. 3, 2002. [Online]. Available: http://www.globus.org/research/papers/anatomy.pdf

[3] "TeraGrid," Web Page, 2001. [Online]. Available: http://www.teragrid.org/

[4] "The Global Grid Forum Web Page," Web Page. [Online]. Available: http://www.gridforum.org

[5] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservation and co-allocation," in *Proceedings of the International Workshop on Quality of Service*, vol. 13, no. 5, 1999, pp. 27–36.

[6] A. Hafid, G. Bochmann, and R. Dssouli, "A quality of service negotiation approach with future reservation (nafur): A detailed study," *Computer Networks and ISDN*, vol. 30, no. 8, 1998.

[7] K. Kim and K. Nahrstedt, "A resource broker model with integrated reservation scheme," in *IEEE International Conference on Multimedia and Expo (ICME2000)*, 2000.

[8] M. Karsten, N. Berier, L. Wolf, and R. Steinmetz, "A policy-based service specification for resource reservation in advance," in *International Conference on Computer Communications (ICCC'99)*, 1999.

[9] J. MacLaren, "Advance reservations: State of the Art," GGF GRAAP-WG, See Web Site at: http://www.fz-juelich.de/zam/RD/coop/ggf/graap/graap-wg.html, Last visited: August 2003.

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

[11] A. Hafid and G.Bochmann, "Quality of service adaptation in distributed multimedia applications," *ACM Springer-Verlag Multimedia Systems Journal*, vol. 6, no. 5, pp. 299–315, 1998.

[12] A. Oguz *et al.*, "The mobiware toolkit: Programmable support for adaptive mobile networking," *IEEE Pesronal Communications Magazine, Special Issue on Adapting to Network and Client Variability*, vol. 5, no. 4, 1998.

[13] G. Bochmann and A. Hafid, "Some principles for quality of service management," Universite de Montreal, Tech. Rep., 1996.

[14] R. Al-Ali, O. Rana, D. Walker, S. Jha, and S. Sohail, "G-QoSM: Grid Service Discovery using QoS Properties," *Computing and Informatics Journal, Special Issue on Grid Computing*, vol. 21, no. 4, pp. 363–382, 2002.

[15] R. Al-Ali, A. Hafid, O. Rana, and D. Walker, "Qos adaptation in service-oriented grids," in *Proceedings of the 1st International Workshop on Middleware for Grid Computing (MGC2003) at ACM/IFIP/USENIX Middleware 2003*, Rio de Janeiro, Brazil, 2003.

[16] I. Foster, C. Kesselman, *et al.*, "The physiology of the grid:an open grid services architecture for distributed systems integration," Argonne National Laboratory, Chicago, Tech. Rep., January 2002.

[17] H. Chu and K. Nahrstedt, “A cpu service classes for multimedia applications,” in *IEEE Multimedia Systems ’99*, 1999.

[18] A. ShaikhAli, O. Rana, R. Al-Ali, and D. Walker, “UDDIe: An extended registry for web services,” in *Proceedings of Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT 2003, IEEE CS Press*, Orlando FL, USA, 2003, pp. 85–90.

[19] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, R. Narayan, and F. Reichmeyer, “Internet2 qbone: Building a testbed for differentiated services,” *IEEE Networks*, 1999.

[20] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, “A Java Commodity Grid Kit,” *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf

[21] N. Zaluzec, “Anl tpm/aaem collaboratory,” See Web Site at: http://tpm.amc.anl.gov/.