Journal of Grid Computing
Editors: P. Kacsuk
Journal no. 10723

Quality of service negotiation for commercial medical Grid services

S.E. MIDDLETON[1], M. SURRIDGE[1], S. BENKNER[2], G. ENGELBRECHT[2]
[1]*IT Innovation Centre, University of Southampton, 2 Venture Road, Chilworth Science Park, Southampton, SO16 7NP, UK {web: www.it-innovation.soton.ac.uk email: sem@it-innovation.soton.ac.uk}*
[2] *Institute of Scientific Computing, University of Vienna, Nordbergstrasse 15, A-1090 Vienna, Austria {web: www.par.univie.ac.at email: sigi@par.univie.ac.at }*

Abstract: The GEMSS project has developed a service-oriented Grid that supports the provision of medical simulation services by service providers to clients such as hospitals. We outline the GEMSS architecture, legal framework and the security features that characterise the GEMSS infrastructure. High levels of quality of service are required and we describe a reservation-based approach to quality of service, employing a quality of service management system that iteratively finds suitable reservations and uses application specific performance models. The GEMSS Grid is a commercial environment so we support flexible pricing models and a FIPA reverse English auction protocol. Signed Web Service Level Agreement contracts are exchanged to commit parties to a quality of service agreement before job execution occurs.
We run four experiments across European countries using high performance computing resources running advanced resource reservation schedulers. These experiments provide evidence for our Grid's rational behaviour, both at the level of service provider quality of service management and at the higher level of the client choosing between competing service providers. The results lend support to our economic model and the technology we use for our medical application domain.

Key words: Grid, quality of service, negotiation, agent, pricing model, economic model

1. INTRODUCTION

The GEMSS project has created a medical Grid for simulation services to be run over the internet. There are many issues involved in creating such a Grid, ranging from the legal and security aspects right through to the quality of service issues and how to negotiate between sets of service providers. This paper presents a brief overview of the GEMSS Grid, implemented and tested at sites in several EU countries, and then focuses on the issues involved in quality of service and the negotiations between competing service providers.

1.1 Paper Objectives

This paper seeks to present the GEMSS Grid for commercial medical simulation services, evaluate our cluster level quality of service negotiation protocol and evaluate our client / service provider(s) negotiation protocol.

1.2 The difference between academic and commercial Grids

Grid technology emerged from academic research, and for some years it has been used to support academic research collaborations, a scenario for which it was originally designed. Systems like Globus [17] allow collaborators to share computing resources, creating a

pool they can all use, which is managed for the common good by a "Virtual Organisation" (VO). In such a Grid, the VO often has a privileged position, trusted by all participants to collect information about resources and to allocate them fairly.

In Grids designed for business, community benefits do not equate to benefits for all its participants, and the balance of risks and rewards for both providers and consumers can be quite different. Three main approaches have been developed to inject commercial business models into the Grid: so-called "Grid economies", Grid-based application service providers (ASP), and business-to-business Grids.

The "Grid economies" maintain the privileged VO approach, but allows resource providers to define the price for using their systems. The Grid-based ASP model can be summarised as a resource-providers' cooperative. Unlike the traditional VO model, consumers do not provide resources to the community, but instead buy services from the collective using a conventional application service provision (ASP) model. Business-to-business Grids follow conventional business practice. The main difference between these and the other models is that the resource providers do not cooperate with each other, nor disclose information to anybody about the availability of their resources. Instead, they wait until customers have a need to perform computations, and then propose a service level agreement.

The GEMSS Grid follows the business-to-business approach, applied to the medical simulation services domain; we enable one to one commercial arrangements between clients and service providers operating under conventional commercial arrangements. The reason behind our requirement for a B2B style Grid is that our end-users (i.e. hospitals) require a commercial arrangement where services are paid for, levels of service expected to be available and penalties demanded if service levels fall below those contractually agreed. Our hospital users also need clear legal agreements with the service providers since personal patient data will be processed and it is the hospital that is required to ensure this data is handled correctly.

1.3 Medical Grid characteristics

The GEMSS project supports six complex medical simulation applications [24], one of which we describe later in more detail. Medical simulation work is characterised by a relatively small numbers of time consuming jobs requiring powerful computational resources (many CPU hours, Gigabyte datasets). We use the Single Photon Emission Computed Tomography (SPECT) application as our exemplar application in this paper. Section 4.1 contains details on the SPECT application.

1.4 Quality of service

Traditional approaches to provide quality of service (QoS) are usually focused on network-level and/or service-level quality of service support, ensuring metrics such as response time and bandwidth are within guaranteed levels. Since high levels of quality of service are required, our Grid infrastructure has been designed to support quality of service guarantees at the application level. Our computing resources have to be reserved in advance to ensure a high availability of the required computing resources; we thus use start time, end time and price as our QoS metrics. In GEMSS we support two reservation capable schedulers - Maui [28] and COSY [9]. Cluster reservations are described using service level agreements, expressed as XML documents following the Web Service Level Agreement (WSLA) [21] specification.

In this paper we refer to this low level QoS negotiation as “microscopic” negotiation.

1.5 Negotiation

A client wishing to run a medical simulation must choose which service provider will run the job. This choice will be affected by the quality of service offered and the price. Our medical end users want to be able to negotiate the best deal before they choose from a set of well-known and trusted service providers. In GEMSS we use agent technology to enable the client to negotiate the best QoS agreement from a set of service providers.

In this paper we refer to this high level client to service provider negotiation as “macroscopic”.

1.6 GEMSS Grid architecture

The GEMSS infrastructure [3] is based on a service-oriented architecture comprising multiple Grid clients and Grid service providers, one or more service registries and a certificate authority (see Figure 1). We only use a single certificate authority in GEMSS since we need maximum control over our user authentication process regarding our sensitive medical data; however if trusted more than one certificate authority could be used.

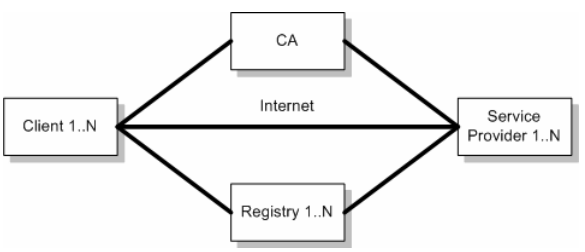

Figure 1 : GEMSS high-level architecture.

The GEMSS infrastructure relies on standard Web Services technologies. GEMSS services are defined via Web Service Definition Language (WSDL) descriptions and

securely accessed using Simple Object Access Protocol (SOAP) messages. For large file transfers SOAP attachments are utilized.

The GEMSS environment supports a three step approach to job execution on the Grid. First there is an initial business step, where accounts are opened and payment details fixed. The pricing model may also be chosen at this stage. Next there is a quality of service negotiation step, where a job's quality of service and price is negotiated and agreed. Finally, once a contract is in place, the job itself can be submitted and executed.

Figure 2 shows the basic architectural components of the GEMSS infrastructure. Whilst the focus in this paper is the QoS negotiation modules, both client and server side, more details on the architecture as a whole can be found in [3]. We do not require special ports (only the usual port 80) be opened in client side firewalls since Grid communication is always initiated by the client; this means end user security department personnel are much more likely to accept our Grid software since its impact on existing network security is low.

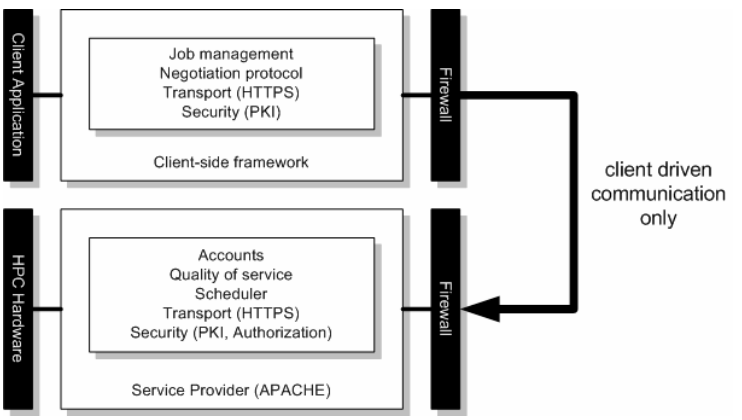

Figure 2 : GEMSS infrastructure.

To provide acceptable levels of quality of service guarantees, well in advance of the job execution time, we adopt a reservation based approach to QoS coupled with application specific performance models; we deploy schedulers that support a reservation mechanism at our service provider sites. To represent quality of service agreements we use the WSLA standard, and implement a reverse English auction protocol for the client / service provider(s) negotiation. Details on all these aspects can be found later in this paper.

We evaluate the GEMSS infrastructure by conducting a series of mirco-negotiation and macro-negotiation tests. The micro-negotiation tests are performed on a 16 CPU cluster running the MAUI scheduler, using a SPECT performance model and SPECT application. The macro-negotiation tests are run using a client application in the UK and

three service providers in Austria, each supporting the SPECT client but using different pricing models.

At a high level we use a three step model to running GEMSS jobs. First a client chooses a business model and agrees contracts with some service providers. Next one or more jobs are negotiated under the terms of the business model. Finally each job is securely executed and results returned. Figure 3 shoes this three step model.

Figure 3. GEMSS three step model

1.7 Paper structure

In section 2 we review related work, discussing work on Grid, negotiation and quality of service. Section 3 presents the business aspects underlying our GRID, including the commercial framework we choose to operator in. In section 4 the quality of service micro-negotiation is introduced, and followed through to section 5 where the client / service provider(s) macroscopic price negotiation is described. Practical evaluation using one of the GEMSS medical applications is reported in section 6 and the results discussed in section 7. We end with conclusions.

2. RELATED WORK

2.1 Business Grids

Examples of academic research collaborations can be found in projects such as the European DataGrid [11] and Enabling Grids for E-sciencE (EGEE) [41]. These projects used Globus and gLite [17] as well as other systems such as Condor to share computing resources as opposed to GEMSS which follows a Web-Service oriented approach. [8] proposed a Grid Architecture for Computational Economy, GRACE (not to be confused with the later EC project of the same name) which provided a generic way to resource parametric sweeping applications controlled by the Nimrod-G resource broker.

An example of a Grid-based ASP model is reflected in the GRASP project [18]. Grid-based ASP's are resourced not by a single provider, but by a whole collective of providers, each of which is paid for pooling their resources and allowing them to be managed by a VO.

The GRIA project [21, 36] is an example of a business-to-business type Grid. This approach does away with a trusted virtual organisation altogether [37], and instead employs business-to-business service provision models that link providers and consumers directly. The GEMSS Grid re-uses work from the GRIA project, specifically technology for access control and resource accounting. It is the combined support for SLA agreements, commercial business models for high value jobs and medical level security that differentiates GEMSS from other Grid architectures.

2.2 Medical Grids

In the bio-medical domain the EU BioGrid Project [4] aims to develop a knowledge grid infrastructure for the biotechnology industry. The main objective of the OpenMolGRID Project [34] is to develop a Grid-based environment for solving molecular design/engineering tasks relevant to chemistry, pharmacy and bioinformatics. The EU MammoGrid Project [27] builds a Grid-based federated database for breast cancer screening. The UK e-Science myGrid Project [31] develops a Grid environment for data intensive in silico experiments in biology.

While most of these projects focus on data management aspects, the GEMSS project focuses on the computational aspect of the Grid, with the aim to provide hardware resources and HPC service across wide area networks in order to overcome time or space limitations of single HPC systems. Other projects in the bio-medical field which also focus more on the computational aspect of the Grid include the Swiss BioOpera Project [6], the Japanese BioGrid Project [22], and the Singapore BioMed Grid [5]. The US Biomedical Informatics Research Network (BIRN) initiative fosters distributed collaborations in biomedical science centred around brain imaging of human neurological disorders and associated animal models. These projects do not address the commercial or security and legal requirements GEMSS considers, since they work with bioinformatics data not sensitive medical patient data.

2.3 Agents and Negotiation

Agent-based systems [23, 39] focus on problem solving entities, embedded into their environment and designed to fulfil a specific role. Agents are autonomous and goal driven, able to work in a reactive or proactive fashion.

Recently there has been a trend [25] to build serious distributed systems based on agent development environments. These architectures (e.g. JADE [2], ZEUS [40], FIPA-OS [14]) provide libraries for agent interaction protocols and follow the various inter-agent communication standards (e.g. FIPA [13], KQML [12]).

The use of Grid with agent technology [15] is relatively new and provides a challenge for the agent development environments. Agent environments tend to provide a closed world, where technologies such as Grid and web services are unavailable. Grids tend to focus on service provision alone. A tighter coupling of agent and Grid technology would allow Grid entities to behave and interact more intelligently, and agent systems to interact with and control meaningful services.

The GEMSS Grid employs the FIPA-OS libraries to implement a reserve English auction between the client and service providers. Adding such agent negotiation technology to a secure Grid system has allowed us to utilize the advantages of both technologies.

2.4 Quality of service

Quality of service has been investigated in various contexts [7, 33] while traditional QoS research usually focuses on network level QoS [38] with various techniques to guarantee service as an improvement to best effort protocols. The work in [7] focuses on QoS support for distributed query processing, providing significant improvements compared to best effort based systems. Our work uses QoS guarantees at the application level to guarantee service runtimes in advance via advance resource reservations. A good discussion regarding the demand for advance reservation in Grid applications can be found in [35] and [29]. Our work focuses on Grid provision of high performance computing (HPC) as opposed to traditional web services.

In [1] a Grid QoS management framework is proposed that mainly focuses on service discovery based on QoS attributes. In [30] a model for QoS-aware component architecture for Grid computing is proposed. The work in [16] proposes a SOAP message tracking model for supporting QoS end-to-end management in the context of Web Service Business Process Execution Language (WSBPEL) and Service Level Agreements (SLA); this paper does not address long running Grid services and SLA negotiation and the trade-off between response time and cost as in our work.

3. ECONOMICS AND BUSINESS MODELS

Studies into the economics of computer services are not new [10]. The computer services market, especially the medical services market, is characterized by well differentiated application vendors, each providing services based around proprietary software; vender lock-in and monopoly situations are not uncommon. In the customers favour, a service market does allow for vertical integration of services, where sub-contractors can work in parallel on different aspects of a task. Demand for computer services is very cyclic in nature, with more demand during the day than there is during the night. The cyclic nature

of demand is in direct contrast with the need for vendors to maximize the utilization of their hardware, which represents a fixed cost to the service provider.

The billing mechanisms for computer services require equitable and auditable charges, but do not necessarily require reproducibility, since this would prevent vendors factoring variables such as demand into their charges. Key to any billing mechanism is the pricing policy. The optimal pricing strategy is considered by [10, 20] to be a flexible one that is free to factor in variables such as system load, application type and specific user groups willingness to pay. Price has a dual role, one of cost recovery (hardware, licensing costs and maintenance costs) and of resource allocation (maximising utilization). Interestingly [10] states that "priority systems are an alternative to a pricing policy and are normally cheaper to implement, however they are simply a surrogate set of prices that may in some instances work as well as a true price mechanism, but will almost never be superior". A number of classic pricing policies are reviewed in [32], including some modern Grid based examples.

3.1 Pricing models

In GEMSS we support a flexible pricing policy that can be customized for each service a service provider supports. The choice of pricing algorithm is left to the service provider, with each pricing model implemented as a simple Java library that can be dynamically plugged in and selected. There are two example pricing models available in the GEMSS initial release, a fixed price telephone pricing model where users are charged at a prearranged CPU hour rate, and a dynamic pricing model where the CPU hour rate is dependant on the current load levels the service provider is experiencing.

We have deliberately kept the complexity of our two pricing models low to reduce the costs involved in both developing the code to compute the price. In a commercial situation we would expect to see initially simple pricing models evolve to more complex and successful models only when the market expands enough to warrant the costs involved (and assuming added complexity yields real profits).

The emphasis in GEMSS has been to explore the issues involved in creating a flexible approach to pricing models, rather than perform a market analysis of the high performance computing / Grid space. Issues such as determining the realistic market rate for each medical applications Grid-based CPU hours will be done after the end of the project in the exploitation phase.

3.2 Accounting model used in GEMSS

The accounting architecture is based on the GRIA [19] accounting model. There is a primary accounts service, which is controlled by the business module on the server side,

and a web interface on the client side. The web interface is intended for both end user budget holders and service provider accounting personnel, allowing access to the current account details and an up-to-date billing statement.

The negotiation handler invokes the business module when a signed WSLA contract is exchanged, and the job handler invokes the business module again when a job has finished executing and its final status (failed or finished) is known. This workflow is shown in figure 4.

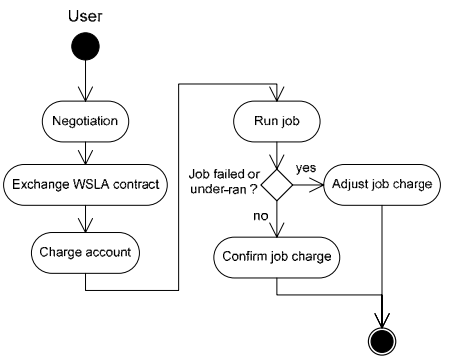

Figure 4 : Accounting workflow.

4. MICRO QUALITY OF SERVICE NEGOTIATION

The micro QoS infrastructure is centred on the QoS manager which provides a high level interface to clients and utilizes the compute resource manager, the application performance model and the chosen pricing model. Figure 5 presents the basic parts of the GEMSS micro QoS infrastructure, which provides the QoS interface to be utilized by the macroscopic QoS negotiation module.

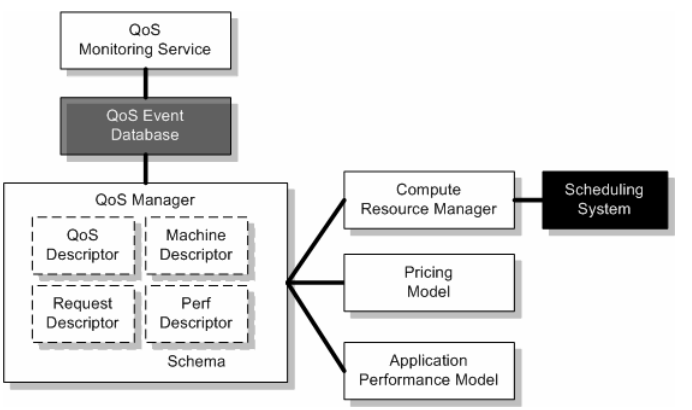

Figure 5 : GEMSS Micro QoS Architecture.

The performance model is used to compute the estimated run time and other performance relevant data for a service request. It takes as input a request descriptor and a machine descriptor and returns a performance descriptor. The request descriptor, supplied by the

client, contains application specific meta-data about a specific service request (e.g. required accuracy). The machine descriptor, supplied by the service provider, specifies the resources that could be offered for an application service (e.g. number of CPUs).

The performance descriptor returned by the performance model usually contains the estimated execution time and other parameters like number of processors used to execute a job, required memory, and required disk space. As a consequence, the performance model may be computed repeatedly until the time constraints are met, varying the number of CPU's etc.

The compute resource manager provides an interface to the scheduler for obtaining information about the actual availability of computing resources (e.g. number of free processors on a machine for a certain time period). The compute resource manager is utilized by the QoS manager in order to check and create temporary reservations during QoS negotiation. Currently a GEMSS resource manager is available for two scheduling systems which provide support for advance reservation, the Maui scheduler [28] and COSY [9].

4.1 Single Photon Emission Computed Tomography (SPECT)

Visualization of the distribution of radio-pharmaceuticals by Single Photon Emission Computed Tomography (SPECT) provides valuable complementary information to the representation of anatomy from high-resolution imaging modalities such as x-ray CT and magnetic resonance imaging. The SPECT application supports fully 3D iterative reconstruction algorithms, providing enhanced image reconstruction for the whole image volume by considering principal 3D effects of data acquisition. A state of the art algorithm – the OS-EM (Ordered Subsets – Maximum Likelihood) algorithm which is an advanced version of the well-known ML-EM (Maximum Likelihood – Expectation Maximization) algorithm is used. This algorithm is based on a stochastic model of Poisson distributed generation and detection of photons. Both, improved resolution and robust convergence criteria are characteristics of this algorithm.

The benefits of fully 3D iterative image reconstruction come with the disadvantage of demanding more computing power than traditional 2D image reconstruction algorithms. Due to the high computational requirements of the reconstruction process, the reconstruction kernel has been parallelized for clusters of shared-memory parallel processors, which are the most suited parallel computer architectures for this application (with a mostly linear speedup as more processors are added). Providing this application as a Grid service enables the use of advanced 3D image reconstruction software for

improved healthcare within a clinical environment (figure 6); the ImageJ plugin is used for the user interface.

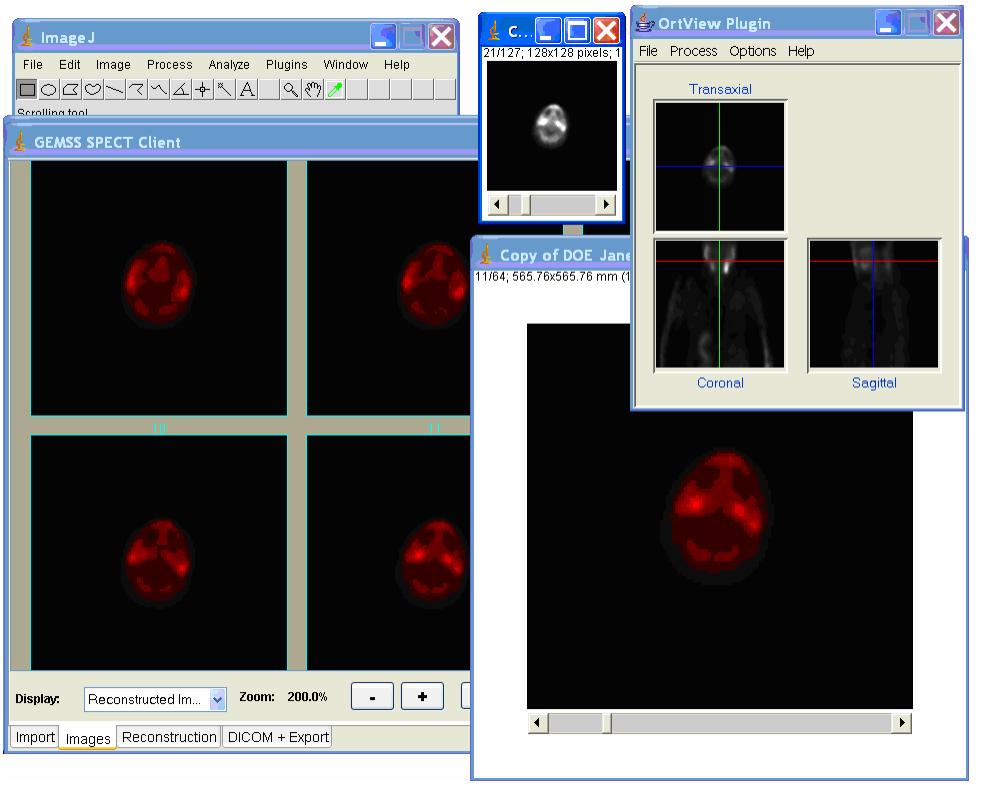

Figure 6 : SPECT application interface

4.2 Performance modelling

The GEMSS infrastructure does not prescribe the actual nature of performance models, since each application is different. Since for many applications it will not be possible to build a simple analytical performance model, the GEMSS micro QoS infrastructure specifies only an abstract interface for performance models. For applications where the provision of an analytical performance model is not feasible, a database could be used to relate typical problem parameters to resource needs like main memory, disk space and execution time, which will initially be populated using data from representative test cases, and which will expand dynamically by including historical data.

For the performance modelling of our SPECT application, we have decided to use an empirical approach to estimate the performance of the reconstruction kernel. First we identify parallel and sequential parts of the reconstruction code and determine the execution time of these parts depending on the input parameters. Next the time for each sequential part is measured by including time marks in the code and logging the output.

By analysing the job output the CPU time needed to run a job, with a specific set of input parameters, can be assessed.

The GEMSS project supported six application models, and we found the analytical models outperformed the database models significantly. The database models required time to gather statistics and large safety margins to ensure reservation windows were large enough (up to 50%) to cope with inaccurate estimations.

4.3 Quality of service manager

The QoS manager relies on heuristics that consider the outcome of the performance model, the availability of resources, and the pricing model to decide whether the client's QoS constraints can be fulfilled. The QoS manager returns a corresponding QoS offer to the client and performs an advance reservation of the required computing resources via the compute resource manager.

The current strategy of the QoS manager is shown in figure 7. The QoS manager executes the performance model and compares the estimated execution time in the resulting performance descriptor with the time constraints of the QoS request. If the client's time constraints can be met, the QoS manager contacts the resource manager to check whether the required resources can be made available. If this is the case, the compute resource manager returns a corresponding resource descriptor, and the QoS manager executes the pricing model. The QoS manager then compares the resulting price descriptor to the client's price constraints. If they can be met, the QoS manager instructs the resource manager to make a temporary reservation of the required compute resources. Finally, the QoS manager generates a corresponding QoS offer, which is returned to the client. If either the performance constraints or the price constraints can not be met, the QoS manager may decide to execute the performance model with a different number of processors (as specified in the machine descriptor). If the client's QoS constraints cannot be met at all, the QoS manager returns the closest offer to the client. In any case the client has to confirm a QoS offer in order that a QoS contract is established.

When a QoS offer is made a temporary reservation is created on the cluster. If the client later fails to confirm this offer (or fails to reply in time) then the reserved CPU time is released back to the scheduler. If the client confirms a QoS offer the client is contractually agreeing to pay for this CPU time, even if the time is later not used (for example the job fails). Other potential business models might provide a refund for a failed job since the service provider can release its reserved resources back to the cluster to run short notice jobs. We have only considered the up-front contractual situation, which was most appropriate for our medial end users requirements.

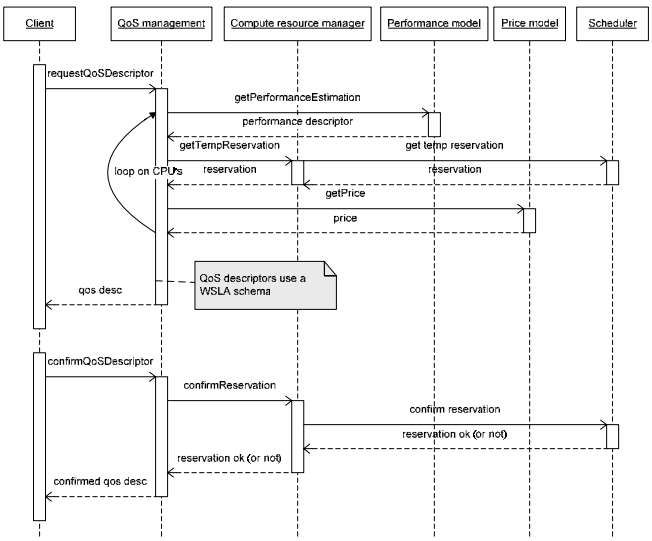

Figure 7 : Micro QoS negotiation.

4.4 Service level agreements

A QoS descriptor is an XML-based document representing a potential agreement on a single service usage between a client and a service provider following the WSLA specification. Being machine readable, using WSLA documents allows us to process the QoS negotiation without the need to human intervention until the final confirmation step. The GEMSS QoS infrastructure utilizes a subset of the WSLA specification. An outline structure of our WSLA document is listed in figure 8.

In the context of GEMSS, SLA parameters are QoS parameters and include the begin time of the job execution, the end time of the job execution, and the price of the job execution. The service definition section specifies the overall contract duration and a metric for each parameter. The obligations section contains a list of objectives. Each objective is linked to an obliged party and defines the acceptable values of a specific SLA parameter.

```
<SLA name="SpectSLA" xmlns="http://www.ibm.com/wsla">
 <Parties>
  <ServiceProvider name="ISC"/>
  <ServiceConsumer name="gerry"/>
 </Parties>
 <ServiceDefinition name="SPECTService">
  ...
  <Operation ... name="start" ...>
   <SLAParameter unit="GMT" type="time" name="beginTime">
   <SLAParameter unit="GMT" type="time" name="endTime">
   <SLAParameter unit="euro" type="double" name="price">
   ...
   <SOAPOperationName>start</SOAPOperationName>
  </Operation>
  ...
 </ServiceDefinition>
 <Obligations>
  <ServiceLevelObjective name="priceObjective">
   <Obliged>provider</Obliged>
   <Validity>
    <Start>2006-09-13T15:23:06.000+02:00</Start>
    <End>2006-09-13T18:10:06.000+02:00</End>
   </Validity>
   ...
   <Expression>
    <Predicate xsi:type="Equal" ... >
     <SLAParameter>price</SLAParameter>
     <Value>33.4</Value>
    </Predicate>
   </Expression>
   ...
  </ServiceLevelObjective>
  ...
  <ServiceLevelObjective name="endTimeObjective">
   ...
   <Expression>
    <Predicate xsi:type="LessEqual" ... >
     <SLAParameter>endTime</SLAParameter>
     <Value>211479006000</Value> <!-- ms since 01/01/2000 -->
    </Predicate>
   </Expression>
   ...
  </ServiceLevelObjective>
 </Obligations>
</SLA>
```

Figure 8 : GEMSS subset of WSLA structure.

5. MACRO QUALITY OF SERVICE NEGOTIATION

In GEMSS we have adopted a practical approach with our Grid middleware, basing our macroscopic negotiation on the Foundation for Intelligent Physical Agents (FIPA) standards. An auction protocol is needed since we have distributed services and want to support flexible pricing policies at each service provider. The reverse English auction protocol is the most appropriate given the GEMSS client/server(s) configuration and firewall restrictions on message passing. We use the WSLA document generated by the micro-negotiation to represent the contract under negotiation.

Our macroscopic QoS negotiation architecture is shown in figure 9, where the modules relevant to the negotiation processes are outlined. The GEMSS Proxy is a proxy class for the service provider's web service interface.

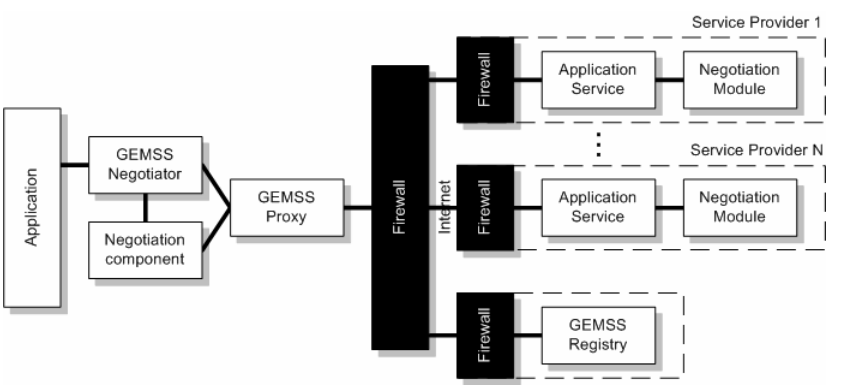

Figure 9 : Macroscopic negotiation architecture.

5.1 FIPA English auction protocol

Our implementation of the FIPA English auction protocol is described in figure 10. A client starts with a job that they wish to execute, and first discovers a set of available services by querying a GEMSS registry. Once a set of suitable service providers is

discovered they are each alerted (informed) to the start of the auction protocol by the client software. The user is asked for the auction criteria, in our case the acceptable min/max values for both the start and end time of the job and the acceptable price range; these are the QoS parameters used in GEMSS. Weightings can be given to each QoS parameter by the user and this information is sent to each service provider via a call for proposals (cfp) message. Each service provider then starts a micro QoS negotiation and comes back with a temporary reservation encoded in a WSLA document. These are collected by the client when the proposal deadline is reached and scored using the scoring algorithm described later in this section. Several rounds of bidding can occur but ultimately a single service provider's WSLA is accepted and the client moved to the job execution phase. One WSLA per service provider per job is agreed – jobs spread over several service providers are not supported, although are theoretically possible.

5.2 Scoring the WSLA

The scoring mechanism used by the GEMSS client is a normalized dot-product score based on a vector of QoS parameter values and a set of parameter 'importance' weighting values. The QoS parameters for our GEMSS test bed are start time, end time and price. The client chooses the importance weights before the auction to reflect the negotiation priorities they want to stress (e.g. the price might be the most important parameter). The QoS parameter set is not restricted and can be easily expanded should a service provider provide other measurable QoS metrics. It is easy to change the scoring mechanism, so more complex approaches tailored to the business environment can be supported in the future.

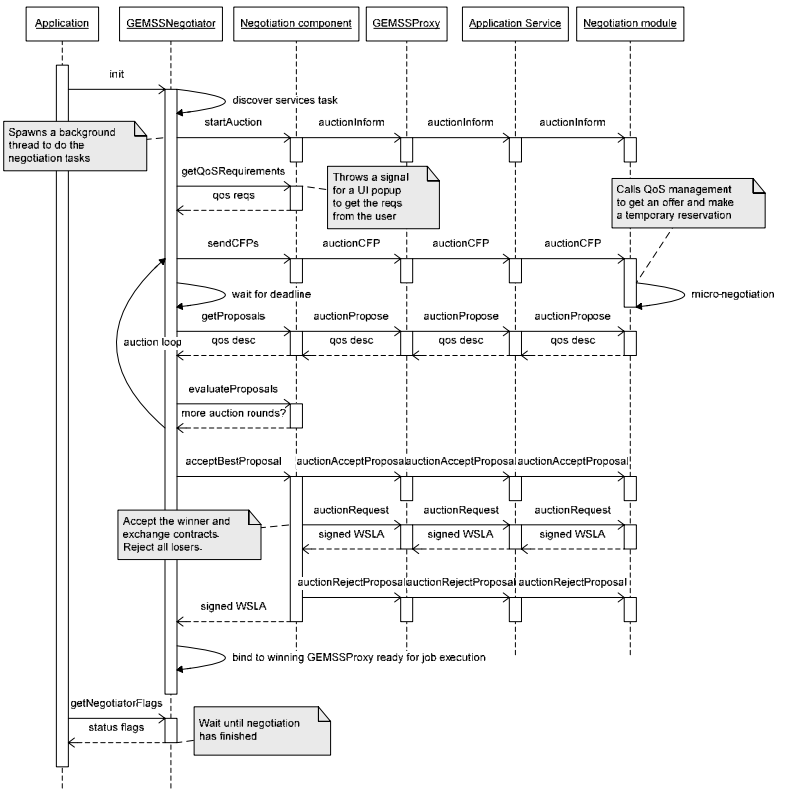

Figure 10 : GEMSS English auction protocol.

6. EXPERIMENTAL EVALUATION

A number of tests have been carried out on the GEMSS infrastructure as part of the evaluation phase of the project. This section details the negotiation evaluation, both at the microscopic and macroscopic levels.

6.1 Evaluation of the microscopic negotiation

The QoS micro-negotiation tests took the form of 53 hours of stress test runs running a total of 1604 real SPECT jobs. All these jobs were run for real on a 48 CPU cluster running a MAUI scheduler. The cluster itself had 3 services set-up to control 16 CPU's each, representing 3 service providers.

For each test 10 client computers were set-up running a single client installation each. Each client concurrently submitted jobs until a total of 401 jobs were submitted. The clients could choose from 3 different kinds of user 'strategy'; one with a high priority for fast job execution, one group with a medium preference for quickly executed jobs and one group with no hard time constraints for the job execution. We wanted a variety of client strategies to simulate real usage where different users will have different priorities.

We identified 24 different SPECT job set-ups to be used in our stress tests, where each job setup has individual input data. These job set-ups represent typical SPECT use-cases

that have been enforced mostly from clinicians and researchers from the GEMSS project. Generally a SPECT job can be characterized with its image resolution, the number of projections acquired from the CT or MRI scanner, the number of slices the should be computed for the output image volume and finally the number of iterations to be computed in order to specify the accuracy of the results. All job set-ups have been categorized by their size accordingly to figure 11 with 8 small, 8 medium and 8 job set-ups as well as a mixture of all 24 job set-ups. Four tests were conducted, where clients ran randomly choosen jobs using a uniform distribution of all available jobs with specified size from figure 11. These four test cases represent different types of stress conditions the cluster can be subjected to.

Size of Job	**Resolution**	**Projections**	**Slices**	**Iterations**
Small	128	60	8 – 32	5 – 25
Medium	128 – 256	60 – 120	64 – 128	5 – 25
Big	256	120	8 – 32	5 – 25
Mixture	Any	Any	Any	Any

Figure 11: SPECT job characteristics

Eval: Micro-negotiation

The metrics measured and results for each of the four stress tests are reported in figure 12. The utilization values represent the percentage of the theoretical full usage of all available nodes and they were reported using the cluster scheduler's own functions, recorded every 15 minutes . Each test attempted to schedule and run 401 jobs. The big job test saw the cluster approaching full load and many jobs were unable to be scheduled due to lack of CPU availability; this reduced the corresponding metric result (i. e. availability heavily decreased).

Stress test metreics

Utilization = Average % node use reported by cluster
Robustness = % of jobs successfully run
Availability = % of successful requests for a SLA
Throughput = Jobs run per hour

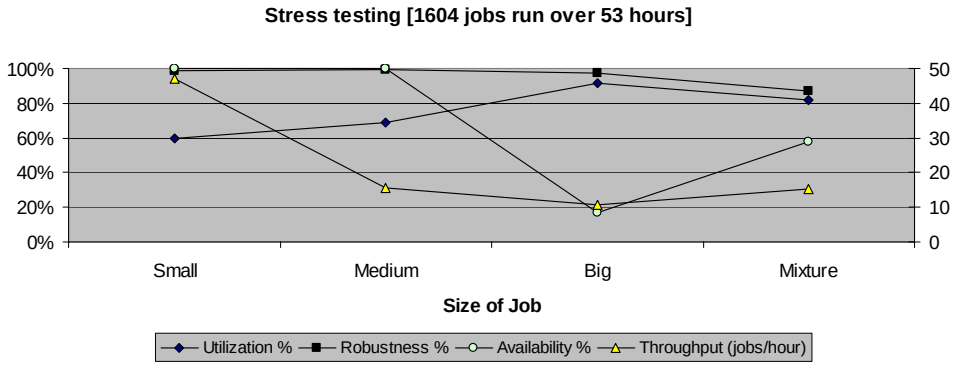

Figure 12 : Stress test results

The runtime of these jobs is dependent on the resources which the jobs are executed on, but in general all jobs could be executed on 2, 4, 8, 12 or 16 nodes on our clusters. Generally jobs with a lower resolution, less number of projections and less slices to compute are performed faster. The overall runtimes on the used resources in our testbed range from a minimum of 2 minutes and a maximum of 90 minutes. Moreover it should be mentioned that bigger jobs could not be executed on less than 4 cluster nodes because of their high memory requirements.

Eval: SPECT performance model

In addition to measuring the stress metrics we recorded the SPECT performance models estimated run time and actual run time for each job we successfully ran. Figure 13 shows these accuracy results. The standard deviation for these figures varied between 0.013 and 0.014 over the four 401 job test runs; these results are statistically significant.

Performance accuracy metrics

Accuracy = (estimated runtime - actual runtime) / actual runtime

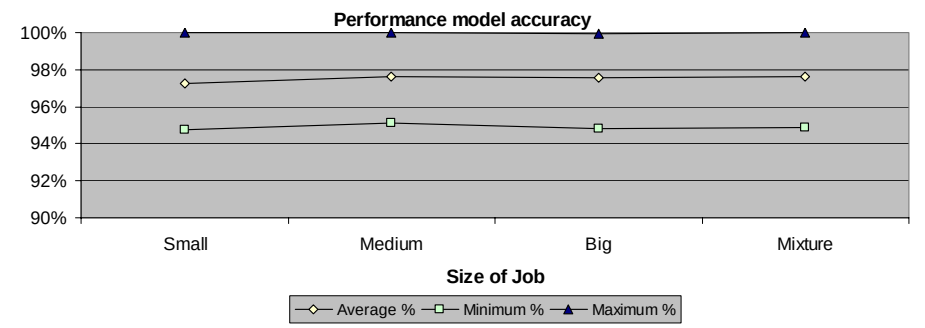

Figure 13 : SPECT performance model accuracy

Analysis

We can see from these tests that the system operates stably over a longer period of time and it is handling most of the requests properly and robustly. The few errors that occurred can be lead back to two different problems that arise throughout the tests. The first problem was related to network timeouts occurring in our hosting environment while querying to the QoS event database (i.e. on status queries) when concurrently other client requests write to this database; we probably had the timeout value too low. Another problem, mainly cased by clients, was that the SLA negotiation could not be performed within the time frame specified by the security token. A solution to this problem would be to extend the validity of the security tokens of the E2E security implementation. In general, however, the system showed few errors even under heavy usage lending evidence of the robustness of our approach.

There were also a number of errors that were handled robustly by the infrastructure, mainly relating to network timeouts whilst negotiating with services. These were handled by an automatic retry mechanism very successfully. Also, if one service was not contactable for some reason the clients simply chose from the other two services that were available.

The cluster's throughput tended to be higher when submitting a lot of small jobs which validates rational behaviour since the smaller jobs are executed faster then bigger ones. The average utilization was lower on the small and medium job sizes because the sizes of these jobs were too small to fully load the cluster. For the big job case the cluster started to reject jobs, being full loaded, and hence availability was reduced. This is to be expected since there is always a trade-off between the client's desire for availability to run new jobs and the service providers desire to keep cluster utilization to a maximum. The theoretical maximum utilization of 100% has been achieved only for the big job sizes, but even tough the average utilization was below the theoretical maximum due to the nature of varying resource requests from clients and possible short periods of unused resources e.g. when a job execution finishes shortly before the reservation interval ends. However, the average utilization level for the big and mixture job cases shows a satisfying demonstration for service and resource providers.

Finally the accuracy of the SPECT performance is very high, verifying that the SPECT application is well suited to this kind of reservation-based model. Other types of less well characterised application are discussed later.

6.2 Evaluation of the macroscopic negotiation

The reason for supporting flexible pricing models is to try to create a practical Grid marketplace where a client can select service providers who will provide QoS guarantees at a reasonable price. Such a market should take into account the service providers need to maximise utilization of their resources and the client's desire for the best price/service. In order to allow us to demonstrate this in action over the GEMSS infrastructure we have devised two tests. The first test runs with a large time window, making start and end time constraints irrelevant; this allows us to analyse the pricing models in isolation and verify rational behaviour. The second test has a smaller window, allowing each service provider's micro-negotiation to schedule jobs where they see fit, within the time and price constraints set by the client. The results of these tests provide an insight into how a real medical service market place might function.

Eval: Macro-negotiation on price alone

Our objective in this test is to demonstrate that the client behaves rationally and chooses the lowest price first. We would expect to see low price models grab the early jobs and quickly fill up the low priced service provider's schedule. If demand is maintained then the clients are forced to pay the higher prices until all service providers are at full capacity, when no more jobs can be run at all.

We installed, in Austria, three service providers as described in figure 14. Each service provider had an individual pricing model of its own. All three service providers provided the SPECT service and ran a MAUI scheduler in simulation mode (to avoid needlessly running the job itself). The SPECT test data was a job that lasted 34 minutes on 2 CPU's.

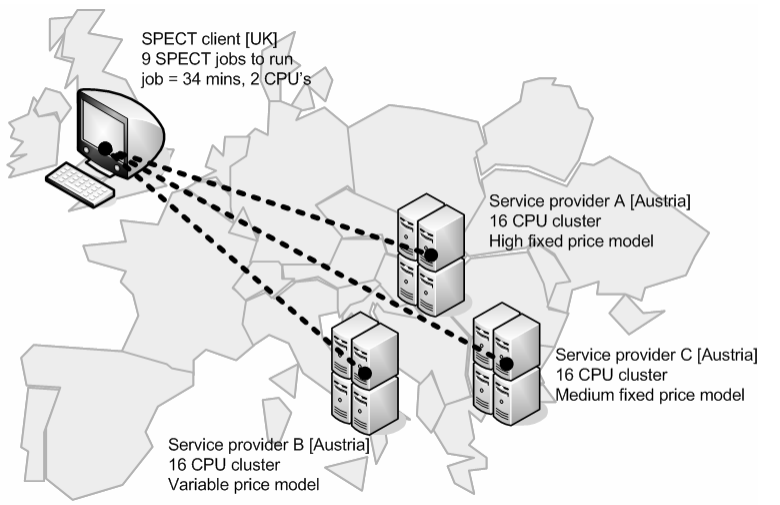

Figure 14 : Test bed deployment.

We set-up an account for the client with each service provider, and credited these accounts with enough euros to run the 9 test jobs. Two fixed price models were used and a variable pricing model that adjusted its price based on the current system load.

To create a controlled environment we manually adjusted the scheduler load to simulate an external cluster load of 9 active CPU's on our 16 CPU cluster. In a commercial implementation the variable pricing model would query the dynamic load directly from its scheduler, or read in a static daily load profile that describes the service provider's cyclic load profile.

Nine jobs were run sequentially, with the client running a negotiation between the three service providers for each job. Once a service provider won a job we increased the providers load by 30%; starting load was 0%. This had the effect of increasing the price demanded by the variable pricing model. We recorded the WSLA QoS parameters (start time, end time and price) and used just a single auction round. Multiple auction rounds are fully supported to allow sophisticated service providers to reduce price in the face of competition, but for this test were not needed. The proposal deadline was 50 seconds, allowing for realistic delays in pan-EU internet communication and firewall protocols; this could be lower with a dedicated high-speed internet link. Inefficient firewalls (and other security) cannot be bypassed since we are dealing with sensitive medical data, and end users will not reduce security of their existing network systems.

The accumulated revenue for each service provider can be seen in figure 15, which shows what happens as each service provider's system load increases (to maximum) and the price for the variable model increases and becomes less attractive.

SPECT test job
Duration 34 mins with 2 CPU's
Price model A
Fixed price = € 5 per CPU hour
Price model B
Variable price = €1 per CPU hour + € 6 * <system load>
<system load> = fractional value [0 .. 1] of cluster's current CPU loading
Price model C
Fixed price = € 3 per CPU hour

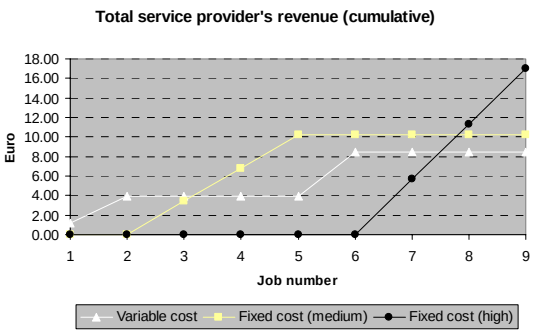

Figure 15 : Revenue from each pricing model (only price variable).

Eval: Macro-negotiation on end time and price

In this test we used the same set-up as test one but allowed the schedulers to vary the start and end time, allowing the schedule to fill up until it was unable to make any more reservations. The client weighted both price and end time equally (weight 1.0), and was not bothered by start time (weight 0.0). The pricing models were the same, except for a new dynamic price shown to make sure it goes above the medium price for an interesting trigger condition. We requested jobs be allocated in a 2 hour window to ensure that a very successful service provider would achieve a full schedule (100% load). For this test we simulated an external loading of 8 CPU's, allowing more scheduling flexibility. For a 2 hour window each job (34 mins) increases the service providers load by about 25%.

SPECT test job

Duration 34 mins with 2 CPU's

Price model A

Fixed price = € 5 per CPU hour

Price model B

Variable price = € 1 per CPU hour + € 10 * <system load>

<system load> = fractional value [0 .. 1]

Price model C

Fixed price = € 3 per CPU hour

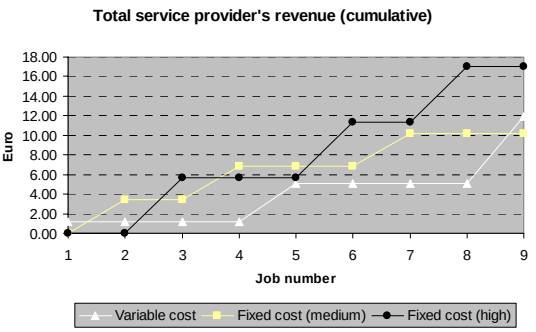

Figure 16 : Cumulative revenues (price, start and end time variable).

Our objective in this test is to examine how our system behaves in a resource limited environment, looking at how demand affects both the success of the pricing policy and the overall resource allocation of the service providers. We would expect to see low prices grab the early jobs and quickly fill up a low price service provider's schedule. Because end time is now important the service providers winning early jobs will no longer be able to finish a job quickly, and hence will become less attractive even with a lower price and thus we should see a more balanced load between service providers.

The results for each job run can be seen in figure 16 and 17, which shows the cumulative revenue gained by service providers and a view of each service provider's reservation schedule as it filled up with jobs.

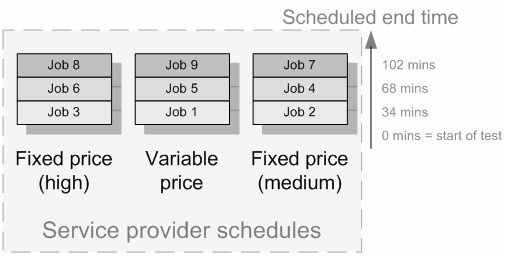

Figure 17 : Microscopic level scheduler reservations.

Analysis

We can see from the two macroscopic negotiation tests that the negotiation results appear to follow rational behaviour. The lowest price job will always get chosen first, and will only be refused when other factors such as the quality of service (e.g. end time of the job) degrade significantly.

If the level of demand is low then the most successful service providers will be the ones offering the lowest price. It can be seen from figure 14 that if only 6 jobs were run (a low demand) then the high fixed price service provider would be very under-utilised since the cheaper providers are chosen first.

If the level of demand is high then the client will be forced to eventually use all service providers, so charging a high price is most sensible. This is all in line with the economic basics of supply and demand.

Limits on resource allocation also allow a high pricing policy to win over because clients are unable to achieve the levels of QoS they need from other loaded providers. Figure 17 shows how each service provider's schedule fills up, and how after one job is allocated on each of the three service providers the best start time available is no longer time 0.

The variable pricing does allow a service provider to get some early cheap jobs in, and defer further load until the client is willing to pay a premium. The exact variable / fixed price ratio could easily be tuned to exploit the thresholds of the fixed pricing models. The optimal variable pricing would maximize both cost recovery to get the revenue in, and resource allocation to ensure the cluster is busy but still able to provide a reasonable QoS.

Regarding scalability the GEMSS architecture supports multiple registries that can theoretically contain 100's of service providers (we had less than 10 in practice). A client can connect to a number of service registries (chosen based on which ones the client trusts) to find all the services providers they wish to deal with. The service provider hosting environments are based on web services and can handle 100's (maybe 1000's depending on hardware) of concurrent requests from clients to reserve CPU's and run jobs.

Given the nature of medical jobs the client (hospital) must have a written legal contract in place before sub-processing of patient data can occur. This legal restriction is actually the limiting factor on how large a medical Grid can scale, not the technology.

7. DISCUSSION

The accuracy of performance models is critical to the ability of the QoS management system to select a large enough reservation to successfully run an applications job. If the accuracy of the performance model is relatively low, for example if the application is fundamentally difficult to predict, the performance model can provide an over-estimate with a large safety margin; this over-sized reservation would be released once the job finish time is known.

We have found that the chosen heuristics of the QoS manager are also crucial to the overall performance of the QoS management. Due to the dynamic nature of resource

allocations for different jobs, and the fact that our flexible pricing model can support different strategies depending on the service provider's preferences, it is very hard to decide which job should be scheduled at which time with a specific number of CPUs involved. Time to fine tune the heuristic parameters should be factored into any future installations.

From our macroscopic evaluation results it can be seen that the existence of fixed and variable pricing models allows service providers to react to the marketplace in different ways. When demand is low the variable pricing model has the flexibility to lower its price to capture early business. This allows a service provider in a low demand market to increase its utilization of resources, which in turns helps cost recovery by maintaining a steady load. The fixed price service providers will have high under-utilization of resource in such a competitive low demand environment. Conversely, in a high demand environment where the demand out-stretches the capacity to meet it we have an effective monopoly situation where clients are forced to run jobs from any service provider who has a few spare CPU cycles to offer. In this case the service provider need only set the price as high as possible, just below the client's maximum threshold before they choose to run no jobs at all. Flexibility is the key here, with the optimal pricing model able to adjust its price based on the levels of demand (most simply measured by scheduler loading level).

A major end user concern at the start of the GEMSS project was that we should stick to good, basic economics and not invent an artificial marketplace with concepts such as Grid credits etc. Keeping tried and tested economic principles in mind has guided us to develop a mechanism for flexible pricing policies, which allow the flexibility for serious future exploitation using pricing models that meet the needs of a real commercial environment. We cannot know what the marketplace will look like in the future, but we can ensure we have the flexibility to adapt to its needs.

To be robust in the face of internet communication delays we used a 50 second auction round duration. This means that it is not practical to negotiate many times for small jobs, since the auction negotiation time would exceed the job exec time. Instead we would expect many small jobs to be bundled together as a batch of jobs and negotiated as a single job with a single contract to be economically viable. On a local network, or high speed dedicated link, negotiation times of around 1 second could be expected making negotiation of smaller jobs more practical.

From an overall viewpoint the use of micro-negotiation for low level scheduler-oriented QoS and macro-negotiation for high level service provider selection appears to

complement each other well. The service provider's micro-negotiation ensures the reservations offered to a client meet the quality of service levels expected by the client and at the same time helps maximize utilization of the service providers computing resources. The client / service provider's macro-negotiation provides the client with a way to compare offers from multiple service providers and maintain a realistic economic model in this global service marketplace.

8. CONCLUSION

The GEMSS project has developed a service-oriented Grid framework that supports the provision of medical simulation services by service providers to clients such as hospitals.

The GEMSS medical applications mostly require computing resources to be pre-booked well in advance of the patient's arrival at the hospital. High levels of quality of service are required, since resource unavailability can result in expensive time being wasted through to compromising the safety of live surgery. The GEMSS infrastructure supports this through a reservation-based approach to quality of service, and provides a microscopic negotiation mechanism where a QoS management system iteratively finds the most suitable reservations available from a resource scheduler based on application specific performance model estimates of each job's resource usage.

The GEMSS Grid works in a commercial environment where clients want to be able to choose from several service providers before agreeing to book a specific resource. The GEMSS Grid supports flexible pricing models for individual services, and a macroscopic negotiation, based on a FIPA reverse English auction protocol, where a client can choose the best offer from a set of competing service providers. A WSLA contract is signed and exchanged to commit both parties before job execution occurs.

We have run four experiments, communicating between two EU countries and using a 48 CPU cluster with a MAUI scheduler, to verify that the microscopic quality of service negotiation and the macroscopic negotiation between client and service providers behave in a rationale manner. The results of this evaluation provide support to our view that the GEMSS Grid provides a realistic economic model, guarantees to clients regarding quality of service, and the basic legal and security framework needed to provide a realistic platform for future exploitation. We hope this work allows us to move closer to being able to set-up a practical commercial Grid for real medical simulation services.

9. ACKNOWLEDGEMENTS

The GEMSS project is funded by the European Commission in the Information Society Technologies (IST) Programme under cross-program theme 'Grid Testbeds' - Framework V Project No. IST-2001-37153

10. REFERENCES

1. Al-Ali, R.J., Shaikhali, A., Rana, O.F., Walker, D.W.: Supporting QoS-based discovery in service-oriented Grids, In Proceedings of the IEEE International Parallel and Distributed Processing Symposium (2003)
2. Bellifemine, F., Poggi, A., Rimassa, G. Jade: a FIPA2000 compliant agent development environment. In Proceedings of the fifth international conference on Autonomous agents, Montreal, Quebec, Canada, 216 - 217 (2001)
3. Benkner, S. Berti, G. Engelbrecht, G. Fingberg, J. Kohring, G. Middleton, S.E. Schmidt, R. Gemss: Grid-infrastructure for Medical Service Provision, In Proceedings of HealthGRID 2004, Clermont-Ferrand, France (2004)
4. BioGrid: The BioGrid Project. http://www.bio-grid.net (2006)
5. BiomedGrid Consortium: http://binfo.ym.edu.tw (2006)
6. BioOpera: Process Support for BioInformatics. ETH Zürich, Department of Computer Science. http://www.inf.ethz.ch (2006)
7. Braumandl, R., Kemper, A., Kossmann, D.: Quality of Serivce in an Information Economy, 2003, ACM Transactions on internet Technolocy, Vol. 3, No. 4, Pages 291-333 (2003)
8. Buyya, R.: Economic-based Distributed Resource Management and Scheduling for Grid Computing, Ph.D Thesis, Monash University, Melbourne, Australia (2002)
9. Cao, J., Zimmermann, F.: Queue Scheduling and Advance Reservations with COSY. In Proceedings of the International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico (2004)
10. Cotton, I.W: Microeconomics and the Market for Computer Services. ACM Computing Surveys (1975)
11. EDG: European Data Grid website at http://web.datagrid.cnr.it. (2006)
12. Finin, T., Mckay, D., Fritzson, R., Mcentire, R. Kqml: An Information and Knowledge Exchange Protocol, in Knowledge Building and Knowledge Sharing, Ohmsha and IOS Press (1994)
13. FIPA: The Foundation of Intelligent Physical Agents, IEEE FIPA Standards Committee, http://www.fipa.org/ (2006)
14. FIPA-OS: source forge project, http://sourceforge.net/projects/fipa-os/ (2006)
15. Foster, I., Jennings, N.R., Kesselman, C.: Brain Meets Brawn: Why Grid and Agents Need Each Other. AAMAS'04, New York, Ney York, USA (2004)
16. Fung, C.K.; Hung, P.C.K.; Guijun Wang; Linger, R.C.; Walton, G.H. A.: Study of Service Composition with QoS Management. In Proceedings of the IEEE International Conference on Web Services, 2005. ICWS 2005, 717 – 724 (2005)
17. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13 (2005)
18. GRASP: The GRASP Project, http://eu-grasp.net/ (2006)
19. GRIA: EC Project IST-2001-33240 Grid Resources for Industrial Applications. http://www.gria.org (2006)
20. Gupta, A., Stahl, D.O., Whinston, A.B.: The Economics of Network Management. Communications of the ACM, Vol 42, No 9, 57-63 (1999)
21. IBM CORPORATION: WSLA Language Specification, Version 1.0. http://www.research.ibm.com/wsla/documents.html (2003)
22. japBioGrid: The Japanese BioGrid Project. http://www.biogrid.jp/ (2006)
23. Jennings, N.R.: An agent-based approach for building complex software systems. Communications of the ACM, 44 (4). 35-41. (2001)
24. Jones, D.M., Fenner, J.W. Berti, G. Kruggel, F. Mehrem, R.A. Backfrieder, W. Moore, R. Geltmeier, A.: The GEMSS Grid: An Evolving HPC Environment for Medical Applications. In Proceedings of HealthGrid 2004, Clermont-Ferrand, France (2004)
25. Luck, M., Mcburney, P., Preist, C.: Agent technology: Enabling Next Generation Computing. AgentLink (2003)
26. Maclaren, J. Sakellariou, R., Krishnakumar, K.T., Garibaldi, J. Ouelhadj, D.: Towards Service Level Agreement Based Scheduling on the Grid. In 14th International Conference on Automated Planning and Scheduling (ICAPS 2004), Whistler, British Columbia, Canada (2004)
27. MammoGrid: The MammoGrid project. http://mammogrid.vitamib.com/ (2006)
28. MAUI: Maui Cluster Scheduler. http://www.clusterresources.com/products/maui/ (2006)
29. Menasce, D.A. Casalicchio, E.: QoS in Grid Computing,IEEE Internet Computing (2004)

30. Musunoori, S.B., Eliassen, F., Staehli, R.: QoS-aware component architecture support for grid. In 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004, 277 – 282 (2004)
31. myGrid: The myGrid Project. http://mygrid.man.ac.uk/ (2006)
32. Nakai, J.: Pricing Computing Resources: Reading Between the Lines and Beyond. NAS Technical Report: NAS-01-010, NASA (2002)
33. Oguz, A., Campbell, A.T., Kounavis M.E., Liao R.F.: The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. IEEE Pesronal Communications Magazine, Special Issue on Adapting to Network and Client Variability, 5(4). (1998)
34. OpenMolGRID: Open Computing GRID for Molecular Science and Engineering. http://www.openmolgrid.org/ (2006)
35. Sulistio, A., Buyya R.: A Grid Simulation Infrastructure Supporting Advance Reservation. International Conference on Parallel and Distributed Computing Systems. San Francisco, CA, USA (2004)
36. Surridge, M., Taylor, S. J., Marvin, D. J.: Grid Resources for Industrial Applications. In Proceedings of 2004 IEEE International Conference on Web Services, pages pp. 402-409, San Diego, USA. (2004)
37. Surridge, M., Taylor, S.J., De Roure, D. Zaluska, E. J.: Experiences with GRIA - Industrial applications on a web services Grid. In Proceedings of 1st IEEE Conference on e-Science and Grid Computing, Melbourne, Australia. (2005)
38. Wang, Z.: Internet Quality of Service, Morgan Kaufmann, San Francisco, CA (2001)
39. Wooldridge, M.: Agent-based software engineering. IEE Proc Software Engineering, 144. 26-37 (1997)
40. ZEUS: source forge project, http://sourceforge.net/projects/zeusagent (2006)
41. EGEE Information sheets, EGEE web site http://www.eu-egee.org/information_sheets