Enabling the Simulation of Service-Oriented Computing and Provisioning Policies for Autonomic Utility Grids

Marcos Dias de Assunção[1], Werner Streitberger[2], Torsten Eymann[2] and Rajkumar Buyya[1]

[1] ***Gr**id Computing and **D**istributed **S**ystems (GRIDS) Laboratory*
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{marcosd, raj}@csse.unimelb.edu.au

[2] *Chair for Information Systems Management*
University of Bayreuth, Germany
{streitberger, eymann}@uni-bayreuth.de

Abstract

A key issue in utility computing environments, such as utility Grids, is the provisioning, orchestration and allocation of resources to services. In these environments, providers need to decide how resources are allocated to service applications according to their workloads, guaranteeing the Quality of Service (QoS) required by customers. Autonomic computing inspired mechanisms are appealing to enable self-organising resource allocation and provisioning. However, these mechanisms are difficult to evaluate in practice either because of the lack of a real test bed or the difficulty in replicating experimental results. This work describes a service framework for a Grid simulator, which enables the modelling and simulation of service-oriented applications. This framework allows the modelling and evaluation of provisioning and negotiation of services and resources. We discuss experimental results that demonstrate the usefulness of the framework for the simulation of a decentralised, self-organising economic model for service and resource negotiation termed Catallaxy.

1. Introduction

Grid computing allows the secure and coordinated sharing of globally distributed resources spanning multiple physical organisations [1]. Service-Oriented Architectures (SOAs) underlie several of the current Grid initiatives and reflect the current Grid computing infrastructures, where participants offer and request application services. A SOA defines standard interfaces and protocols that enable developers to encapsulate resources of different complexity and value as services that clients access without knowledge of their internal workings [2, 3].

Grid systems have therefore increasingly been structured as networks of interoperating services that communicate with one another via standard interfaces. Scientists can provide data, algorithms and applications as services to other members of the scientific community. In addition, with the advent of utility computing environments, several resource providers host services and provide the tools needed by scientists and companies to expose the core functionalities of their research or business as services that are subsequently used by clients or collaborators.

In utility computing environments resource providers offer resources to host services in a pay as you go fashion. There is a clear separation of providers and consumers in such environments. Virtualisation technology offers powerful resource management mechanisms for utility computing by enabling performance isolation, migration, suspension and resumption of virtual machines. One key issue, however, is the provisioning, orchestration and allocation of resources to services. Utility computing providers need to decide how resources are allocated to service applications according to their workloads, guaranteeing the QoS expected by their customers. Autonomic computing [4] inspired mechanisms and policies are appealing to enable self-organising allocation of resources to services in these environments, as well as for service provisioning and negotiation [5-8].

However, it is challenging to design and evaluate practical allocation policies that permit utility computing environments to self-manage and adjust resource

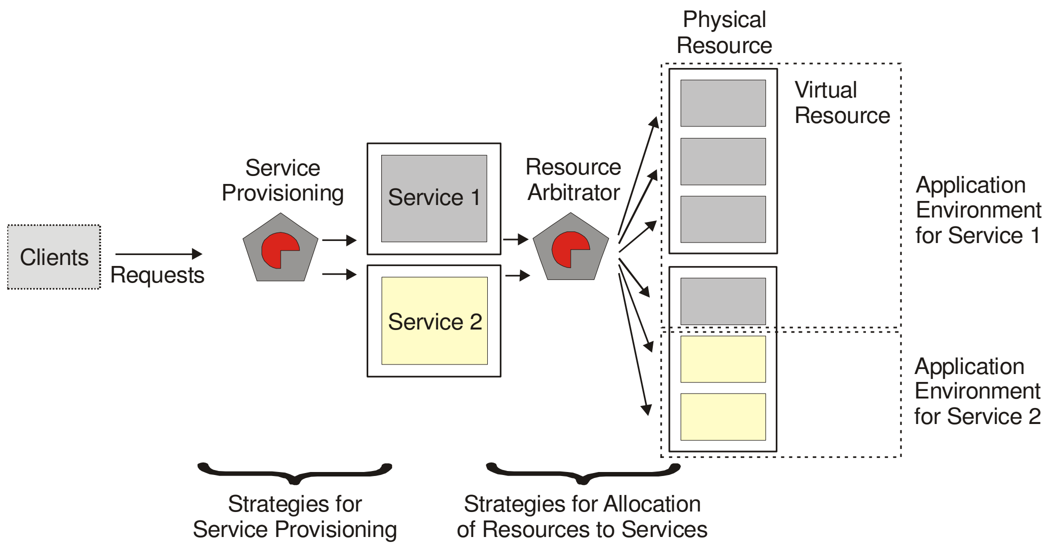
Physical Resource
Virtual Resource
Application Environment for Service 1
Application Environment for Service 2
Service Provisioning
Resource Arbitrator
Service 1
Service 2
Clients
Requests
Strategies for Service Provisioning
Strategies for Allocation of Resources to Services

Fig. 1 – Abstract view of a utility data centre.

allocations according to the provisioning decisions of the offered services. Moreover, it is also a challenge to evaluate these policies and negotiation strategies either due to the difficulty of replicating experiments or a lack of a real test bed. The modelling and evaluation of these mechanisms and related policies can be augmented by the use of simulators.

Even though Grids and data centres are moving towards such a utility and autonomic computing scenario, simulation tools do not keep up and still focus on issues related to resource modelling and allocation assuming in general a job abstraction. The existing Grid simulation toolkits do not provide the features needed to model and simulate services, their placement on resources, their workloads and provisioning policies let aside the abstraction of containers or virtual machines.

In this work, we present a framework that allows the modelling, simulation and evaluation of mechanisms and policies for service provisioning, negotiation and resource management. The framework supports the simulation of service-oriented applications, and considers service dependencies, for different domains including high-performance, on-demand and utility computing. We demonstrate the usefulness of our framework by modelling and simulating an Application Layer Network (ALN) and an economic model for service and resource negotiation termed Catallaxy.

Therefore, the main contributions of this work are to:

- Provide a framework for modelling and simulation of service-oriented applications and autonomic policies for service provisioning and resource orchestration in utility computing environments.
- Demonstrate the usefulness of the service framework by modelling and evaluating an economic model for service provisioning and resource allocation for ALNs.
- Present empirical results that show the usefulness of the Catallaxy economic model for resource allocation and service negotiation.

The remaining part of this paper is organised as follows. Section 2 presents background and related work. Section 3 describes the service framework. In Section 4, we present the design of a decentralised economic bargaining model for ALNs (i.e. the Catallaxy). Section 5 presents the performance evaluation results and finally, Section 6 concludes the paper.

2. Background and Related Work

In this section, we present an illustrative scenario to evidence the mechanisms and policies that we would like to model and simulate. We consider a utility data centre that hosts service applications and also provides resources on demand to its customers' business applications. A simplified model of the utility data centre is presented in Fig. 1 [9]. From right to left, the centre is composed of a pool of physical resources that are managed by server virtualisation technology [10, 11]. The services offered to customers run on Application Environments (AEs) within the resource pool, which are isolated from one another. An AE is a set of virtual resources, that is, containers or Virtual Machines (VMs). The resource arbitrator allocates resources to each AE according to the resource allocation policies in order to meet the required performance and service levels.

In this scenario, customers can utilise services without the knowledge of the internal infrastructure of the resource layer and the resource allocation policies. However, customers and providers negotiate the Quality of Service (QoS) required, and customers want to have guarantees about the service delivery. These guarantees are negotiated and established through Service Level Agreements (SLAs). Service provisioning policies define how the service is provided in order to achieve the service levels stated in the SLA. In this case, the provider has to decide on how the service is provisioned.

The services have a workload that can vary. The number of requests to the hosted services and the expected QoS levels will guide the arbitrator on the resource allocation decisions. The arbitrator has to decide the resources required by each service and whether new resources have to be allocated to meet demand peaks or not. A decoupling of service and resource layers allows one to model strategies for the placement of services on resources and resource orchestration. One can also evaluate distinct markets or mechanisms for service negotiation and resource allocation. Therefore, a provider in this scenario has two policies: one that defines how a service is provisioned and one that defines how resources are allocated.

This is naturally an example; however, a simulation framework should be flexible enough to enable the modelling and simulation of varying scenarios. For instance, the ALN presented in this work follows a two layer market model. In one layer, resource providers provide processing and storage resources. Service providers negotiate with resource providers to acquire capacity to host services. The second layer corresponds to the negotiation between service providers for the delivery of composite services. For example, a service provider can negotiate the access to several atomic services in the service market to deliver it as a bundle, or composite service, to its customers. Similar scenarios are considered in other utility computing strategies [12].

2.1 Related Work

Several Grid simulators allow the modelling and simulation of Grid resources and allocation policies; examples include OptorSim [13], SimGrid [14] and MicroGrid [15]. OptorSim provides the features needed to model and evaluate the data transfer and replication strategies in data Grids. It is a discrete event simulator implemented in Java and follows the abstraction of data resources. The main goal of this simulator is to provide a means for evaluation of data transfer strategies in a data Grid, and so it does not provide a service-oriented application model.

MicroGrid enables the emulation of a Grid environment. A user can run her Grid application on this emulated environment, while the simulator intercepts the exchanged messages. Although it is possible to simulate service-oriented applications, MicroGrid does not provide a decoupling of the service and resource layers that would allow the design and evaluation of different strategies or economic mechanisms for each layer.

SimGrid provides a set of abstractions and functionalities that can be used to build simulators for several application domains. The core functionalities can be used to model and evaluate parallel application scheduling on distributed computing platforms. SimGrid also provides emulation facilities for running distributed and parallel applications in an emulated Grid environment. SimGrid is a trace based event simulator and, like the simulators previously described, uses the abstraction of 'resources'.

GridSim [16] is a Java-based Grid simulation toolkit that provides features for application composition, information services, and the ability to model heterogeneous computational resources of variable performance. In addition, GridSim provides an auction framework that allows the design and evaluation of auction protocols for Grid systems. By using these features, it is possible to model and evaluate the scheduling of jobs on Grid resources and evaluate the impact of the allocation policies. GridSim has the features necessary to design and model the resource layer.

The features provided by GridSim enable the modelling and simulation of intricate Grid environments. However, it does not provide a service framework for simulating service-oriented Grid applications. In this work we opted to leverage the existing features of GridSim and provide a service framework that enables the modelling and evaluation of service provisioning policies, resource allocation policies and multiple economic mechanisms for service negotiation and resource management. GridSim, along with the extensions described here, provides means for evaluating autonomic computing systems, utility computing environments and utility Grids.

3. A Service Framework for GridSim

This section discusses the service framework for simulating service-oriented Grid applications in

GridSim. First, we describe the requirements for simulating service applications. Next, we present how the architecture and the framework fulfil these requirements. Finally, we present an example of the usage of the service framework.

3.1 Requirements for the Service Framework

A framework for simulation of service-oriented Grid applications needs to satisfy the following requirements:

Clear decoupling of resources and services: Services require resources to serve clients' requests. However, there should be a separation between the service and the resource layers. This allows one to model and evaluate different strategies for each of the layers. For example, a decentralised bargaining economic model can be used to allocate resources. However, once the resources are obtained by a service provider to host the services, the provider can engage in a centralised market, in which providers and clients place offers and bids for service usage.

Separation of service provisioning from resource acquisition policies: In addition to the distinction between service and resource layers, a provider's policies also have to be split into two groups, namely provisioning and acquisition policies. Provisioning policies define how the provider decides and negotiates on the allocation of services to clients, while the acquisition policies define the provider's behaviour when negotiating and obtaining resources or atomic services required to deliver a composite service.

Service information repositories for resource and service discovery: Clients or providers can query repositories or Peer-2-Peer (P2P) networks for discovering services. When querying, clients and providers can specify the characteristics as well as the cost of the services they need.

Negotiation and bargaining for the provision of services and resources: The framework has to support means for one to model and evaluate negotiation and bargaining models for services and resources. The negotiation model has to be generic enough so that a negotiation can take place between a client and a provider or between two providers. Several negotiation models can be modelled and evaluated.

3.2 Realisation of a Service Framework for GridSim

In this section, we firstly describe GridSim and then present the service-framework justifying some design decisions. We present a high-level description of key concepts regarding the model and main components of GridSim. For a thorough explanation, we refer to previous work [16, 17].

GridSim is a discrete event simulator built on top of SimJava2 simulation package. A simulation in GridSim comprises of GridSim entities that communicate with one another by passing and scheduling simulation events. GridSim adopts a job model, that is, applications are modelled as jobs or tasks that are executed on Grid resources. A *Gridlet* corresponds to a job, which has parameters like the job length expressed in Millions of Instructions (MIs), the amount of CPUs required, among others. It is possible to model Grid resources of varying configurations such as supercomputers, commodity clusters and personal computers. The processing capacity of a resource's CPUs is expressed in Millions of Instructions Per Second (MIPS). GridSim provides resource allocation policies such as space-shared, time-shared and space-shared supporting advance reservations. However, the user may implement her own resource allocation policy by extending the abstract class *AllocPolicy*, defining how a resource's CPUs are allocated.

GridSim provides a hierarchical Grid Information Service (GIS) that can be comprised of multiple regional GISs. At the start of the simulation, a Grid resource registers itself with a regional GIS. A user can define what information a Grid resource should provide to the GIS; however, by default the Grid resource registers only its ID and whether it supports advance reservation or not. In addition, GridSim allows the modelling and simulation of data resources and catalogues for data Grids, and network topologies. The features listed above allow a user to model resource brokers and varying scheduling strategies for Grid computing.

In a utility computing environment resource providers have resource pools and provision these resources to service providers or consumers to host service applications. The service providers need to acquire resources to host the service applications to be able to serve customers' requests. Service providers provision their services to customers and are willing to provide a given QoS under a given number of requests. The service requests will impose a workload on the resources allocated. The provision of a composite service may require the use of other atomic services.

Thus, the proposed framework considers two distinct stages: (i) the negotiation for and allocation of the resources to host services, and the negotiation for services and the required QoS; and (ii) the actual utilisation of the services and resources. The framework provides means for modelling service registries and discovery, service and resource negotiation as well as means for measuring the resource utilisation imposed by the services' workloads.

A provider in this scenario has two policies: one that defines how a service is provisioned and one that defines how resources are allocated to a service. With the advent of server virtualisation, the allocations may change according to the service workloads.

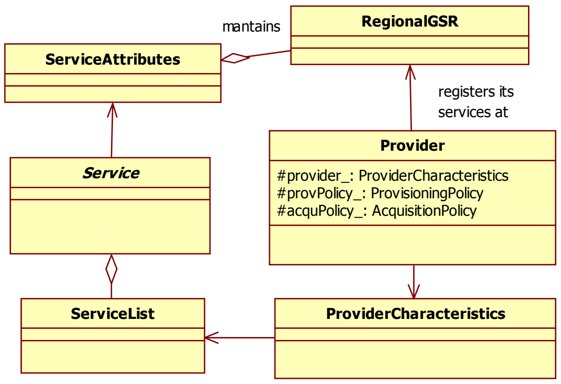
mantains
RegionalGSR
ServiceAttributes
registers its services at
Service
Provider
#provider_: ProviderCharacteristics
#provPolicy_: ProvisioningPolicy
#acquPolicy_: AcquisitionPolicy
ServiceList
ProviderCharacteristics

Fig. 2 - Relationship between Provider, Service and RegionalGSR.

We term the policy that defines how a service is provisioned Provisioning Policy while the allocation policy is termed Acquisition Policy. The class *Provider* is a GridSim entity that implements the basic behaviour for a provider. A provider has characteristics represented by *ProviderCharacteristics*. The class *ProviderCharacteristics* contains a list of *Services* offered by the provider and other attributes like time zone, and the provisioning and acquisition policies utilised. A *Service* corresponds to a service offered by the provider and has *ServiceAttributes* and *ServiceRequirements*. At the start of the simulation, the provider registers itself and the attributes of her services with a regional Grid Service Registry (GSR). *ServiceAttributes* include information like service cost, name and type. We opted for implementing service attributes as a distinct class for the sake of performance and minimisation of simulation events. The *ServiceRequirements* correspond to atomic services or specific resources required to deliver the service to clients. For example, a provider may offer a service, but does not allocate resources to it until the service is required. Fig. 2 demonstrates the relationship between services, providers and GSRs.

The Provider can engage in a market with clients for negotiating its resources. It can also participate in different markets with different mechanisms for negotiating and providing the resources necessary to host the services and satisfy the requests for a service.

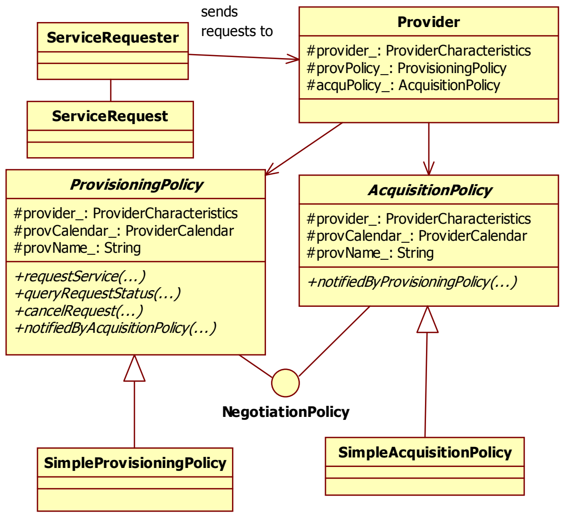
ServiceRequester
sends requests to
Provider
#provider_: ProviderCharacteristics
#provPolicy_: ProvisioningPolicy
#acquPolicy_: AcquisitionPolicy
ServiceRequest
ProvisioningPolicy
#provider_: ProviderCharacteristics
#provCalendar_: ProviderCalendar
#provName_: String
+requestService(...)
+queryRequestStatus(...)
+cancelRequest(...)
+notifiedByAcquisitionPolicy(...)
AcquisitionPolicy
#provider_: ProviderCharacteristics
#provCalendar_: ProviderCalendar
#provName_: String
+notifiedByProvisioningPolicy(...)
NegotiationPolicy
SimpleProvisioningPolicy
SimpleAcquisitionPolicy

Fig. 3 - Class diagram for the provider and the policies.

Fig. 3 shows a diagram describing providers and the basic provisioning policies implemented. Both *ProvisioningPolicy* and *AcquisitionPolicy* implement the *NegotiationPolicy* interface. A *NegotiationPolicy* defines the methods necessary to handle negotiations for service provisioning or resource allocation based on WS-Agreement. The provisioning policy defines how the provider manages the negotiation with clients for service provisioning and how it handles the resource requests. The acquisition policy specifies the provider's behaviour in negotiating with other providers for accessing the required services or resources. These services can be needed for composite services and the resources are required to host service applications. In many instances, provisioning and acquisition policies have to be synchronised or informed about one another decisions, as demonstrated by Grit *et al.* [18]. We provide methods that allow the policies to be synchronised.

Two examples of provisioning and acquisition policies are provided. In the provided implementation of a provisioning policy, *SimpleProvisioningPolicy*, the provider accepts requests while the maximum number of instances for the service is not achieved. The acquisition policy, *SimpleAcquisitionPolicy*, selects the first resource from the provider's resource pool to deal with the workload generated by the service requests.

Although the *Provider* class can be extended, it is not necessary since it is possible to define different behaviours for a provider by extending the *ProvisioningPolicy* and *AcquisitionPolicy* classes to provide the strategies required.

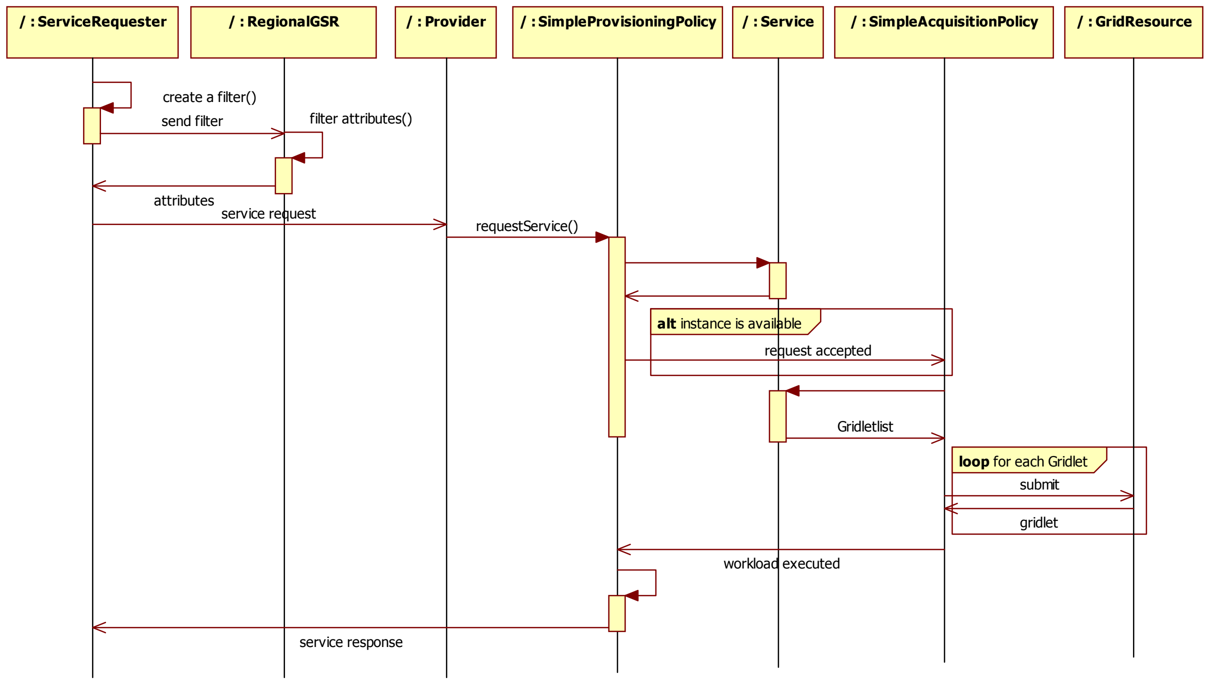

Fig. 4 - An illustrative interaction of a service invocation.

The *ServiceRequester* class is a GridSim entity that can query services at a GSR and make requests to providers. These queries can be performed by passing a filter to the GSR, which corresponds to specifying the parameters for a query. For example, the service requester can pass an object whose class extends *ServiceFilter* to select all the *ServiceAttributes* with a given service type and name. The GSR uses the filter to select and return a list of *ServiceAttributes* that match the given criteria.

A request for a service accepted by a provider generates a workload. The workload is composed of items that can be either requests for atomic services or *ServiceGridlets* that are sent to the resources allocated to the service. The *ServiceGridlet* class extends *Gridlet* by specifying additional parameters such as memory and storage required to fulfil the request. The values of these parameters for a service request can be estimated through profiling techniques, such as those described by Urgaonkar *et al.* [19], where a service application is examined in isolation and its workload is obtained by analysing the use of resources such as memory, CPU and disk. By following this model it is possible to analyse the impact of different provisioning and acquisition decisions on resource utilisation.

3.3 Modelling a Service-Oriented Grid Application

Fig. 4 presents an interaction diagram that illustrates a simple example of the use of the service framework. We consider that providers are assigned a number of services and have already registered the service attributes with a regional GSR. A service requestor then starts by creating a filter and asking the regional GSR to send a list of service attributes that match the given criteria. When the list is returned by the GSR, the service requester selects a provider and requests the service. For the sake of simplicity, we do not consider negotiations for service usage in this example. Once the provider receives the service request, it decides whether to accept the request or not based on its provisioning policy. If the number of instances for that service has not reached the maximum number of instances, the provider accepts the request. When the provider accepts the request, the acquisition policy is notified, so that it can allocate the resources needed to host the service and execute its workload. The service workload is obtained in this case by passing the *ServiceRequest* to the service. As demonstrated in more detail in Section 7, the implementation of the service returns the workload by considering several parameters of the request, such as input file size and expected use time for the service. The service workload is a list of *WorkloadItems*, in this case *ServiceGridlets* that need to be executed on Grid resources. The *SimpleAcquisitionPolicy* allocates the resources needed to execute the *ServiceGridlets* from the provider's resource pool. Once the resources have been allocated, *SimpleAcquisitionPolicy* sends the *ServiceGridlets* to them and monitors their execution. Once all *ServiceGridlets* complete execution, the acquisition policy will notify *SimpleProvisioningPolicy*, which will in turn inform *ServiceRequester* that the execution of the service request has been finished.

Although the acquisition and allocation of resources in this illustrative scenario is made after a request is accepted, this is generally not the case. A different implementation of the *AcquisitionPolicy* can define that service provider should reserve a set of resources in advance, place the service applications on them, and based on the resources available, take the decisions regarding the provisioning of services.

4. The Catallaxy Scenario

The CATNETS project, funded by the European Union, investigates the use of an economic model, termed Catallaxy, for service negotiation and resource allocation in ALNs, such as Grids and P2P networks. This section describes the conceptual applicability of the GridSim service framework to the catallactic economic model utilised in CATNETS.

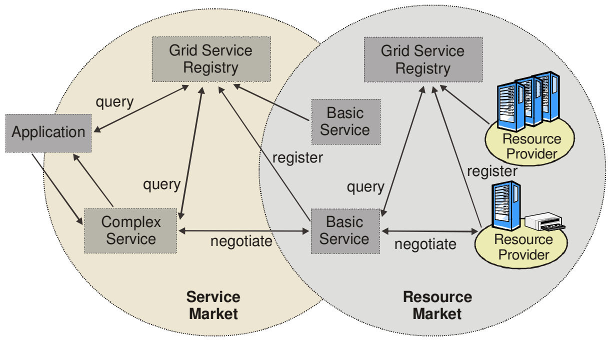

Fig. 5 - Catallaxy market model.

Catallaxy is a decentralised self-organising economic model derived from Hayek's concept of spontaneous order [20]. The Catallaxy concept is based on the explicit assumption of self-interested actions of the participants, who try to maximise their own utility and choose their actions under incomplete information and bounded rationality [21]. The goal of Catallaxy is to achieve a state of coordinated actions, through the bartering and communication of members, to achieve a common goal that no single user has planned. Hayek's Catallaxy concept is the result of descriptive, qualitative research about economic decision-making of human participants. Its results are taken literally to construct ALN markets with software participants, who reason about economic decisions using artificial intelligence.

Algorithm 1 AcquisitionPolicy

```
1:  repeat forever
2:    event ← wait for an event
3:    if event = message from provisioning policy then
4:      proposals ← Ø
5:      request_accepted ← the request ∈ event
6:      cfp ← create cfp for request_accepted
7:      send cfp
8:      proposals ← collect the proposals
9:      best ← select best proposal ∈ proposals
10:     start bargaining process
11:     outcome ← result of the bargaining process
12:     if outcome = success then
13:       inform other participants about the success
14:     end if
15:     apply learning algorithm
16:     notify provisioning policy about outcome
17:   end if
18:   if event = learning message then
19:     treat message received
20:     apply learning algorithm
21:   end if
```

Fig. 6 – Pseudo-algorithm of the execution of an acquisition policy.

In ALNs, the participants offer and request services and compute resources of different complexity and cost. The interdependencies between services and resources are split by creating two interrelated markets: a resource market for trading of computational and data resources; and a service market in which the trading of services takes place. This separation allows instances of a service to be hosted on different resources [22]. Fig. 5 shows the abstract model adopted by CATNETS. A Complex Service (CS) is a composite service, like a workflow, that requires the execution of other interdependent services, termed Basic Services (BSs). A CS is the entry point for the application layer network. The traded products on the service market, the BSs, are completely standardised and have a single attribute name. The name is a unique identifier whose intended semantics is shared among all complex service providers. Multiple instances of the same BS can co-exist in the ALN. For example, two or more basic service providers are allowed to provide a specific BS.

The service market is used by Complex Service Providers (CSPs) to allocate BSs from Basic Service Providers (BSPs). BSPs are registered in a GSR. A CSP queries a GSR to receive a list of required trading partners (BSPs) able to provide the BS required. This list is ranked according to the BS offered price. The best

BS offer is selected for the succeeding bargaining process. This discovery process is modelled using GSRs and discovery process offered by the simulation framework.

Algorithm 2 ProvisioningPolicy

```
 1: repeat forever
 2:   event ← wait for an event
 3:   if event = call for proposals then
 4:     cpf ← get the call for proposal ∈ event
 5:     proposal ← formulate proposal for cfp
 6:     reserve the resources
 7:     send proposal
 8:   end if
 9:   if event = bargaining then
10:     start bargaining process
11:     outcome ← result of bargaining process
12:     if outcome = success then
13:       notify acquisition policy
14:       inform other participants about the success
15:     else
16:       release resources
17:     end if
18:     apply learning algorithm
19:   end if
20:   if event = reject proposal then
21:     release the resources
22:   end if
23:   if event = learning message then
24:     treat message received
25:     apply learning algorithm
26:   end if
```

Fig. 7 – Pseudo-algorithm of a provision policy.

After a successful negotiation in the service market, BSPs negotiate with Resource Providers (RPs) for the resources necessary to host services and serve the service requests. RPs utilise the existing resource management systems to allocate the necessary resources. RP offer resources in resource bundles. A resource bundle is described by a set of pairs of resource type and quantity. Every BS has an associated resource bundle. The bundle defines the type and quantity of resources needed for provisioning that service. In the CATNETS scenario, the resource bundle required for a BS is predefined for the sake of simplicity. In general, the model allows the use of any BS to resource bundle mapping function. In the resource market, the allocation process follows the service market. First, a BSP queries for RPs which are able to provide the specified resource bundle and ranks the received list of RPs according to the offered price. Second, the bargaining for the resource bundle is carried out. If the resource negotiation ends successfully, the BS is executed on the contracted resources from a RP.

To realise these two markets in GridSim, we have implemented provisioning and acquisition policies for the three kinds of providers, namely CSPs, BSPs and RPs. The providers differ in terms of the policies used for service and resource provisioning and acquisition. The execution of the market participant's policy for acquiring services or resources (i.e. **AcquisitionPolicy**) is shown in the pseudo-algorithm in Fig. 6 and those of a market participant's policy for service provisioning (i.e. **ProvisioningPolicy**) is depicted in Fig. 7.

The most important part of the implemented policies is the utilised bidding strategy. This includes what a provider bids. The bid denotes the provider's valuation and reservation prices, i.e. the maximum price which an agent is willing to pay for the service and the minimum price an agent has for selling a BS or a resource bundle respectively. The generation of the valuation is influenced by external factors such as the market price and the learning algorithm. For the formal model of the implemented strategy we refer to the work by Reinicke *et al.* [23].

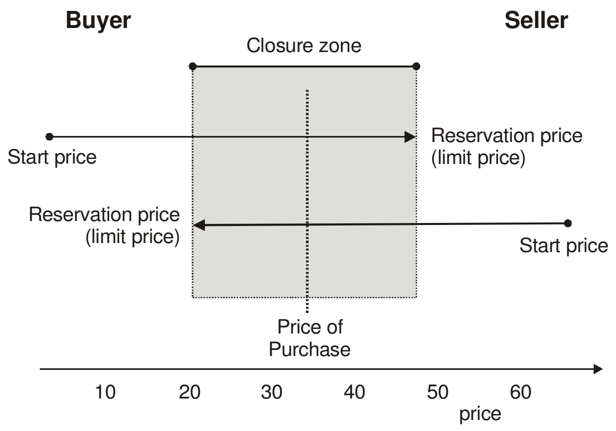

Fig. 8 - Bilateral negotiation process.

The proposed realisation for the CATNETS markets is the usage of a bilateral negotiation protocol for exchanging bids in a point-to-point communication. The initial situation is depicted in Fig. 8. Both trading partners define a reservation price that reflects their estimation of the value of the good. For a buyer, this is the maximum price; for a seller it is a minimum price. The start price represents the negotiation starting point. By subsequent concessions, the opponents move closer to a compromise and a possible contract. Each opponent tries to maximise its own utility, which is the difference between the price of purchase and the reservation price. Thus, buyer and seller policy converge to a trade-off point in an iterative way using the exchange of offers and counter-offers and successive concessions. Rosenschein and Zlotkin [24] call this a monotonic concession protocol. For each iteration, the policy implementation chooses the best of three possible actions for the next communicative act to be made: accept offer, propose counter-offer (with concession or otherwise), or reject offer.

In the implementation of the CATNETS scenario using GridSim, CSs and BSs are modelled as Services. The service requirements of a CS define the BSs needed to deliver the CS. The service requirements of a BS define a minimum resource bundle required to host the BS. The requirements of a BS *j* are represented by $BSR_j = (u_j, p_j, y_j, m_j, s_j)$, where u_j is the number of resources required; p_j represents the number of CPUs in each resource; y_j is the speed of the processors in MIPS; m_j is the amount of memory per resource; and s_j represents the storage capacity required.

Algorithm 3 receiveCFP(cpf_j)

```
1:  $BSR_j$ ← obtain required resource bundle from $cfp$
2:  $RB_i$ ← the resource bundle advertised
3:  $selected\_resources$ ← Ø
4:  $booking\_id$ ← 0
5:  for each resource $R_i \in RB_i$ do
6:    if $R_i$ is not allocated then
7:      if $p_j \le p_i$ and $y_j \le y_i$ and $m_j \le m_i$ and $s_j \le s_i$ then
8:        $selected\_resources$ ← $selected\_resources \bigcup R_i$
9:      end if
10:   end if
11:   if $selected\_resources = u_j$ then
12:     $booking\_id$ ← book($selected\_resources$)
13:     break for
14:   end if
15: end for
16: if $booking\_id \neq 0$ then
17:   $proposal$ ← create_proposal($selected\_resources$)
18:   send($proposal$)
19: else
20    reject($cpf_j$)
21: end if
```

Fig. 9 – RP's strategy upon the arrival of a CFP.

A RP has a resource pool within which it creates Application Environments (AEs) with the resource configuration required by a BS. A resource bundle corresponds to the resources offered by the RP. A resource bundle *i* is represented by $RB_i = (u_i, p_i, y_i, m_i, s_i)$, where u_i is the number of resources in the bundle; p_i represents the number of CPUs in each resource; y_i represents the speed of the processors in MIPS; m_i is the amount of memory per resource; and s_i represents the storage capacity per resource. A RP registers the bundle with the GSR, which is viewed as a service by the BSP. That is, RP provides a service that consists in allowing the BSP to acquire resources.

As described beforehand, once a negotiation for a BS finishes in the service market, a negotiation for the resources needed for the BS starts at the resource market. The BSP will search for RPs that can provide a resource bundle that has the minimum amount of resources required. The BSP will then start the negotiation by sending a Call For Proposals (CFP) to the selected RPs. The RPs whose resources have not been allocated, will formulate a proposal. Once the bargaining process is finished, the RP will allocate its resources to host the BS. Although a RP can divide its resource pool in various ways and change the allocations of AEs over time, in the CATNETS implementation we consider that they are pre-determined and do not change. We also consider that a RP can allocate only part of its bundle to an AE to host a BS when the BS does not require the whole bundle. The strategy followed by a RP when it receives a CFP from a BSP for the negotiation of resources for a BS is summarised in the algorithm presented in Fig. 9.

The RP examines its resource bundle to check whether there are resources available to host the BS. If the resources are available, RP will lock the resources and will send a proposal. Although not included in this algorithm, if RP receives a reject proposal message, the resources will be released. We have also omitted the process of formulating the proposal.

5. Performance Evaluation

In this section we present experimental results that demonstrate that GridSim with the extensions discussed in this work can be used to model and evaluate service provisioning and resource allocation policies for service-oriented Grids, and autonomic utility computing environments. The experiments particularly measure how the Catallaxy model, built on top of the discussed framework, coordinates the use of services and resources. We evaluate the allocation rate by identifying the number of service requests that are satisfied and the overhead imposed by the service and resource negotiations.

5.1 Experimental Scenario

We consider an environment in which RPs provide resource bundles and BSs require a particular resource bundle for a given time slot to host the service and execute the service workload. The experiments have been carried out considering a CS termed Workflow Service (WFS) that requires two BSs, namely Processing Service (PS) and Storage Service (SS). These two BSs, in turn, require a unit of Processing Bundle (PB) and a unit of Storage Bundle (SB) respectively. A PB offered by a RP is composed of multiple resources. Each resource in PB corresponds to exactly one unit of PB required by a PS. A resource in a PB has the following configuration: (p = 2, y = 1500MIPS, m = 1GB and s = 2GB). A resource in a SB is given by: (p = 1, y = 1500MIPS, m = 2GB and s = 4GB).

TABLE I
DESCRIPTION OF PARAMETERS USED IN THE EXPERIMENTS

Acronym	Parameter
PWS	Providers of Workflow Services
PPS	Providers of Processing Basic Services
PSS	Providers of Storage Basic Services
PPB	Providers of Processing Resource Bundles
PSB	Providers of Storage Resource Bundles
SI	Service Instances Per WFS Provider
RU	Resource Units Per Resource Provider
WSR	Requests to Workflow Service
TBWS	Time between arrivals of WS requests
WSRL	WFS Request Length
PSRL	PS Request Length
SSRL	SS Request Length
INSIZE	Input File Size
OUTSIZE	Output File Size

TABLE II
VALUES FOR THE PARAMETERS FOR THE DIFFERENT EXPERIMENTS

Parameter	Exp. 1	Exp. 2	Exp. 3	Exp. 4
PWS	10	20	50	50
PPS	10	20	50	50
PSS	10	20	50	50
PPB	10	20	50	20
PSB	10	20	50	20
SI	40	40	40	40
RU	1	1	1	1
WSR	1000	1000	1000	1000
TBWS	0~120s	0~120s	0~120s	0~120s
WSRL	30~60s	30~60s	30~60s	30~60s
INSIZE	30~50KB	30~50KB	30~50KB	30~50KB
OUTSIZE	100~200KB	100~200KB	100~200KB	100~200KB

We perform our experiments with varying numbers of RPs, BSPs and CSPs. The parameters used in the experiments are shown in TABLE I. The values for PS Request Length (PSRL) and SS Request Length (SSRL) are given by *WSRL / 2* because we consider that WFS first requires processing and further stores the results of the processing activity. For simulating the workload of PS and SS and obtaining the final time of the service utilisation, we consider a simple approach. For example, the workload generated by an invocation j of PS at RP i is given in MIs by: $WPS_j = p_j * y_j * PSRL_j$ where p_j is the number of processors required by the PS, y_j is the processor speed in MIPS and $PSRL_j$ is RS request length.

5.2 Experimental Results

TABLE II describes the experiments performed and the values used for the simulation of the service application in GridSim using the Catallaxy economic model and the presented service framework. The parameters TBWS, WSRL, INSIZE and OUTSIZE use uniform distributions. We consider that the BSPs are able to provide and negotiate for one BS at a time.

Fig. 10 shows the allocation rate in the different experiments. The allocation rate is above 96% in all experiments. However, in Experiment 3 the allocation rate is lower than in Experiment 4, even though more resource providers are available. The reason for such behaviour is that a provider reserves its services or resources when it receives a CFP. Once an announcement is sent by the provider who initiated the negotiation, the providers that have not been selected release their services or resources. As the number of providers increase, more messages are sent, the negotiations take more time and the resources are kept reserved for a longer time. In Experiment 4 we reduce the number of resource providers and determine that the allocation rate increases.

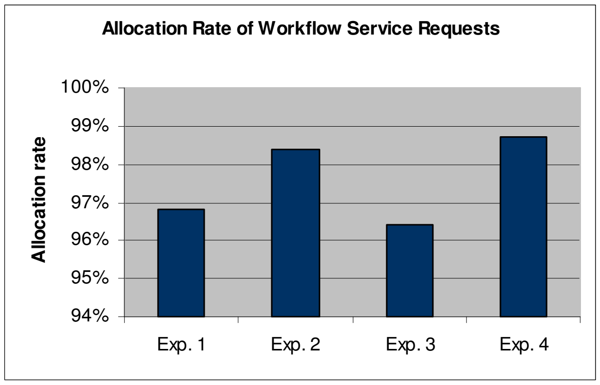

Fig. 10 - Allocation rate of Workflow Service requests.

We then evaluate the impact of the negotiations on the service provisioning process. The experiments measure the amount of time spent on negotiation for a BS. Fig. 11 shows the time spent in different scenarios. We observed that the time spent is highly dependent on the initial timeout during which the negotiator waits for proposals, which in this case is 30 seconds (15 seconds in negotiation for the BS and 15 seconds in negotiation for the resource). We omitted this 30 second interval from the results presented in the figure. In the scenarios evaluated, we consider that users and service providers are in different networks connected through a network link with a bandwidth of 1Mbps while service providers and resource providers are connected through another network link with a bandwidth of 1Mbps. Both links present a latency of 50 milliseconds, which we consider to be representative of the latency in many wide area networks.

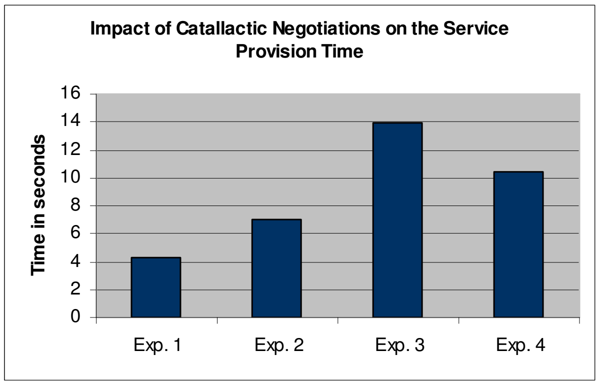

Fig. 11 - Amount of time spent in a Catallactic negotiation for a basic service.

The time required to send proposals and to bargain to achieve the final price is generally smaller than 10 seconds. The initial timeout can be reduced if the initial negotiator knows how many providers have been contacted and how many messages should be received. However, we envision a scenario in which a P2P network is used to broadcast calls for proposals and the negotiator does not know exactly how many providers will receive the proposals and send a reply.

6. Conclusion and Future Work

This paper describes a model for simulation of service-oriented Grid applications, allowing the decoupling of service negotiation and resource management into two distinct layers. By decoupling these into distinct layers, it is possible to model and evaluate different strategies for both service provisioning and resource allocation. The model also allows the simulation and evaluation of policies for negotiation of SLAs for service usage. Evaluation of centralised and decentralised economic models is enabled by extending the provisioning and acquisition policies provided by our framework.

We present experimental results that demonstrate the use of the framework for modelling and evaluation of a decentralised economic bargaining mechanism, the Catallaxy, for service and resource negotiation.

As future work, we would like to evaluate the suitability of the framework for modelling large-scale scenarios and improve the acquisition policies to support advance reservation and co-allocation of Grid resources. In addition, we would like to evaluate the economic models considering dynamic environments with varying failure probabilities for resources. We will consider acquiring data from existing Grid test beds for determining the failure probability of Grid resources and include these in the Grid simulator.

In addition, we would like to incorporate models for what can be called *elastic* containers or *elastic* VMs. In these scenarios, the allocation policy of a utility data centre, for instance, may decide for expanding the amount of memory, storage and CPU of VMs in an AE according to the service workload. We would like to incorporate these VM models and enable the changes in the configurations of VMs on the fly. These features can enable the evaluation of varying provisioning policies.

Acknowledgment

We thank Anthony Sulistio, Krishna Nadiminti and James Broberg from the University of Melbourne for their help in extending GridSim and sharing their thoughts on the topic. This work is supported by the European Union, DEST and ARC Project grants. Marcos' PhD research is partially supported by NICTA.

References

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[2] I. Foster, "Service-Oriented Science," *Science,* vol. 308, pp. 814-817, 2005.

[3] M. P. Singh and M. N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*: John Wiley & Sons, Ltd., 2005.

[4] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," in *Computer*. vol. 36: IEEE Computer Society Press, 2003, pp. 41-50.

[5] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian, "Resource Management in the Autonomic Service-Oriented Architecture," in *3rd IEEE International Conference on Autonomic Computing (ICAC 2006)*, Dublin, Ireland, 2006, pp. 84-92.

[6] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen, "Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure," in *3rd IEEE International Conference on Autonomic Computing (ICAC 2006)*, Dublin, Ireland, 2006, pp. 5-14.

[7] M. N. Bennani and D. A. Menascé, "Resource Allocation for Autonomic Data Centers using Analytic Performance Models," in *2nd IEEE International Conference on Autonomic Computing (ICAC 2005)*, Seattle, Washington, 2005, pp. 229-240.

[8] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goya, "Dynamic Provisioning of Multi-tier Internet Applications," in *2nd IEEE*

International Conference on Autonomic Computing (ICAC 2005), Seattle, Washington, 2005, pp. 217-228.

[9] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility Functions in Autonomic Systems," in *International Conference on Autonomic Computing (ICAC 2004)*, 2004, pp. 70-77.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, 2003, pp. 164-177.

[11] P. Fabian, J. Palmer, J. Richardson, M. Bowman, P. Brett, R. Knauerhase, J. Sedayao, J. Vicente, C.-C. Koh, and S. Rungta, "Virtualization in the Enterprise," *Intel Technology Journal,* vol. 10, pp. 227-242, August 2006.

[12] C. Low and A. Byde, "Market-Based Approaches to Utility Computing," Internet Systems and Storage Laboratory, Hewlett Packard Laboratories Bristol, Technical Report HPL-2006-23 February 2006.

[13] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini, "Simulation of Dynamic Grid Replication Strategies in OptorSim," in *3rd International Workshop on Grid Computing (GRID 2002)*, London, UK, 2002, pp. 46-57.

[14] H. Casanova, "Simgrid: a toolkit for the simulation of application scheduling," in *1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia, 2001, pp. 430-437.

[15] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, "The MicroGrid: a Scientific Tool for Modeling Computational Grids," in *ACM/IEEE Supercomputing 2000 Conference (SC'00)*, Dallas, USA, 2002, pp. 53-53.

[16] A. Sulistio and R. Buyya, "A Grid Simulation Infrastructure Supporting Advance Reservation," in *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, MIT Cambridge, Boston, USA, 2004, pp. 1-7.

[17] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," in *The Journal of Concurrency and Computation: Practice and Experience (CCPE).* vol. 14, 2002, pp. 1175–1220.

[18] L. Grit, D. Inwin, A. Yumerefendi, and J. Chase, "Virtual Machine Hosting for Networked Clusters: Building the Foundations for 'Autonomic' Orchestration," in *1st International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006) - held in conjunction with SC06*
Tampa, Florida 2006.

[19] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," in *5th symposium on Operating systems design and implementation*, Boston, Massachusetts, 2002, pp. 239-254.

[20] F. A. v. Hayek, *The collected works of F.A. Hayek.* Chicago: University of Chicago Press, 1989.

[21] H. A. Simon, *Models of Man - Social and Rational*. New York: John Wiley & Sons, 1957.

[22] T. Eymann, O. Ardaiz, M. Catalano, P. Chacin, I. Chao, F. Freitag, M. Gallegati, G. Giulioni, L. Joita, L. Navarro, D. G. Neumann, O. Rana, M. Reinicke, R. C. Schiaffino, B. Schnizler, W. Streitberger, D. Veit, and F. Zini, "Catallaxy-based Grid Markets," *International Journal on Multiagent and Grid Systems, Special Issue on Smart Grid Technologies & Market Models,* vol. 1, pp. 297-307, 2005.

[23] M. Reinicke, W. Streitberger, and T. Eymann, "Scalability Analysis of Matchmakers in Self-Optimizing Computing Networks," *Journal of Autonomic and Trusted Computing (JoATC),* 2005.

[24] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. Cambridge: MIT Press, 1994.