Lehrstuhl für Wirtschaftsinformatik

Information Systems Management

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Floriano Zini (ITC-irst Trento), Gianfranco Giulioni (Universita delle Marche Ancona), Michael Reinicke, Werner Streitberger, Torsten Eymann (Universität Bayreuth)

Analysis of Simulation Environment

Bayreuth Reports on Information Systems Management

UNIVERSITÄT BAYREUTH

Authors:

Torsten Eymann (University of Bayreuth)
Michael Reinicke (University of Bayreuth)
Werner Streitberger (University of Bayreuth)
Floriano Zini (ITC-irst Trento)
Gianfranco Giulionio (Universitá delle Marche Ancona)
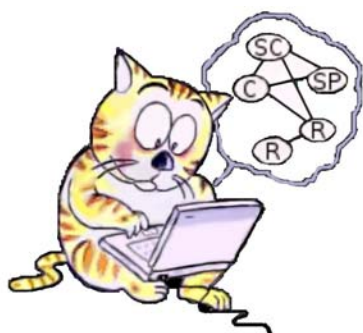
IST-FP6-003769 CATNETS

D2.1
Analysis of Simulation Environment

Contractual Date of Delivery to the CEC: 31.08.2005	
Actual Date of Delivery to the CEC: 13.09.2005	
Author(s): Floriano Zini (ITC-irst Trento), Gianfranco Giulioni (Università delle Marche Ancona), Michael Reinicke, Werner Streitberger (Universität Bayreuth)	
Workpackage: WP2-Simulation	
Est. person months: 9	
Security: public	
Nature: final	
Version: 1.0	
Total number of pages: 45	

Abstract:

This document analyses the requirements for the CATNETS simulation environment and evaluates a number of grid and general purpose simulators; a reasoned choice of a suitable simulator is performed.

Keywords (optional):

CATNETS simulator, requirements analysis, simulator selection.

CATNETS Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-FP6-003769. The partners in this project are: LS Wirtschaftsinformatik (BWL VII) / University of Bayreuth (coordinator, Germany), Arquitectura de Computadors / Universitat Politecnica de Catalunya (Spain), Information Management and Systems / University of Karlsruhe (TH) (Germany), Dipartimento di Economia / Università delle Marche Ancona (Italy), School of Computer Science and the Welsh eScience Centre / University of Cardiff (United Kingdom), Automated Reasoning Systems Division / ITC-irst Trento (Italy).

University of Bayreuth
LS Wirtschaftsinformatik (BWL VII)
95440 Bayreuth
Germany
Tel: +49 921 55-2807, Fax: +49 921 55-2816
Contactperson: Torsten Eymann
E-mail: catnets@uni-bayreuth.de

University of Karlsruhe
Institute for Information Management and Systems
Englerstr. 14
76131 Karlsruhe
Germany
Tel: +49 721 608 8370, Fax: +49 721 608 8399
Contactperson: Daniel Veit
E-mail: veit@iw.uka.de

University of Cardiff
School of Computer Science and the Welsh eScience Centre
University of Caradiff, Wales
Cardiff CF24 3AA, UK
United Kingdom
Tel: +44 (0)2920 875542, Fax: +44 (0)2920 874598
Contactperson: Omer F. Rana
E-mail: o.f.rana@cs.cardiff.ac.uk

Universitat Politecnica de Catalunya
Arquitectura de Computadors
Jordi Girona, 1-3
08034 Barcelona
Spain
Tel: +34 93 4016882, Fax: +34 93 4017055
Contactperson: Felix Freitag
E-mail: felix@ac.upc.es

Università delle Marche Ancona
Dipartimento di Economia
Piazzale Martelli 8
60121 Ancona
Italy
Tel: 39-071- 220.7088 , Fax: +39-071- 220.7102
Contactperson: Mauro Gallegati
E-mail: gallegati@dea.unian.it

ITC-irst Trento
Automated Reasoning Systems Division
Via Sommarive, 18
38050 Povo - Trento
Italy
Tel: +39 0461 314 314, Fax: +39 0461 302 040
Contactperson: Floriano Zini
E-mail: zini@itc.it

Changes

Version	Date	Author	Changes
0.1	05.08.05	Floriano Zini	First draft
0.2	31.08.05	Floriano Zini	Second draft
1.0	13.09.05	Floriano Zini	Final version

Contents

Chapter 1

Introduction

One of the main issues in the development of future Grid and Peer-to-Peer (P2P) network systems is the efficient exploitation of services and resources available in the network. A scalable, dynamic, and adaptable allocation mechanism is essential so that *quality of service* requirements from network users are met. For example, imagine a distributed system for on-demand provision of digital music. The typical user of such a system would like to rapidly and economically access music-playing services which, in turn, must be able to rapidly retrieve low-cost digital music files in order to meet users' desiderata. In this scenario there is the need of service and resource allocation mechanisms that are able to allocate both music playing services to users and digital music files to playing services taking into account the continuous changes in network service load and placement of music files.

The overall objective of the *CATNETS* project is to determine the applicability of a decentralized economic self-organized mechanism for service and resource allocation to be used in *Application Layer Networks (ALN)*. The concept of ALN is an abstraction that integrates different overlay network approaches, like Grid and P2P systems, on top of the Internet. Their common characteristic is the redundant, distributed provisioning and access of data, computation or application services, while hiding the heterogeneity of the service and resource network from the user's view.

The allocation mechanism to be evaluated is based on the economic paradigm of *Catallaxy* [HBKC89, EP00]. The term Catallaxy can be seen as a synonym of "market economy", where participants in the market work for their own good and utility. In our case participants are users, service managers, and resource managers which operate in the ALN. The research question to which we want to answer is if Catallactic service and resource allocation mechanisms will lead to superior ALN performance measured using economic parameters.

Evaluation of Catallactic allocation mechanisms is planned to be performed via simulation. Simulation is a common and useful technique for the analysis, design and evaluation of computer systems [Jai91] and is largely used in distributed computing for the

analysis of network systems. Its main advantage is low cost evaluation of the system while performing a thorough exploration of the possible operative scenarios in a controlled way. If the system is not available yet, as it is often the case during the design stage, simulation can be used to predict the performance and/or compare several design alternatives. Even if the system is available, a simulation model allows for its evaluation under a variety of workloads and environments.

Evaluation will be conducted by comparing the behaviour of the Catallactic mechanism with other allocation approaches in simulated ALNs under a wide range of operative scenarios. This will allow detailed investigation of aspect such as scalability, topology influence, or connection reliability. In particular, the various comparisons and evaluations we want to perform are illustrated in Figure 1.1. The four cells of the matrix contain differ-
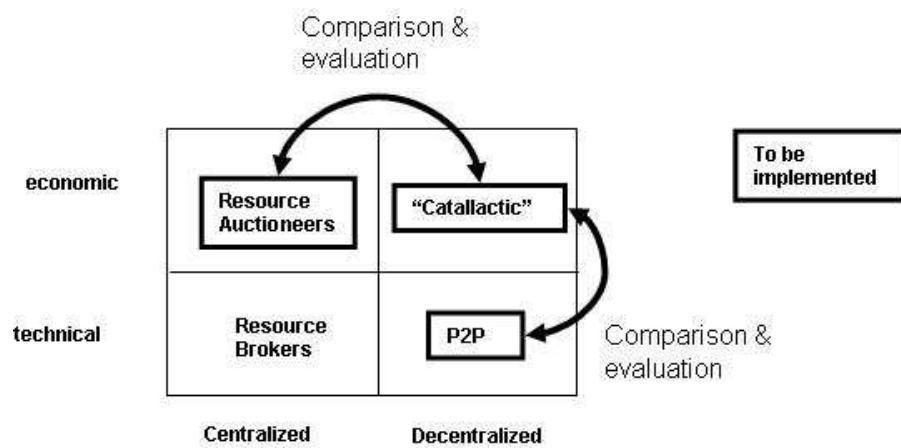
Figure 1.1: Evaluations/comparisons to be performed via simulation.

ent approaches to service/resource allocation. The Catallactic approach (upper-right corner) is both economic and decentralised and will be firstly compared and evaluated with respect to an economic but centralised mechanisms (see arrow at the top of Figure1.1).

A second interesting type of experiment is the evaluation/comparison of the catallactic approach with non-aconomic allocation mechanisms used in P2P networks (see arrow at the right of Figure 1.1). Even though it would be desirable to perform such a comparison, it is not in the central focus of the project and it will be given lower priority.

In order to be able to perform extensive and thorough experimentation, we need a simulation tool which allow for the modelling of ALN scenarios at the "right" level of detail, the plug-in of the various service/resource allocation mechanisms to be evaluated, and the variation of the set of parameters that are necessary for realistic simulation. Therefore, the selection of simulation tool for our experiments that is suitable as it is or easily modifiable for our purposes is fundamental.

The first part of this deliverable analyses which are the salient features of the ALN scenarios to be simulated and the requirements for a simulator for such scenarios. Then,

the document describes some grid and general purpose simulators we think can be reasonable candidates for CATNETS and assesses them with respect to the requirements previously elicitated. The motivated choice of the simulation tool is then presented and the description of a preliminary enhancement plan for the chosen simulator concludes the report.

Chapter 2

Simulator Requirements

This chapter reports the requirement analysis for the CATNETS simulator. The analysis guided the process of simulator evaluation and selection presented in Chapter 4. We start our analysis by identifying two fundamental non-functional requirements that need to be always taken into consideration in the simulator evaluation and selection process.

1. The simulator should be based upon up-to-date and well-supported technology that is well-known by the consortium members;

2. The past experience of the consortium members in the development and use of specific simulation tools are key factors which can speed-up the development of the CATNETS simulator.

Given the two premises above, we identify two main aspects to be taken into account for the elicitation of the requirements for the CATNETS simulator.

ALN modelling. We need simulations which are independent from specific instances of Application Layer Networks. The CATNETS simulator should allow for the specification of ALN models which do not depend on specific types of P2P or grid systems but are abstract enough so that we can derive from their execution general conclusions about performance of service and resource allocation mechanisms.

Scalability. We need to simulate scenarios where there are ALN with a relevant number of nodes and/or a high number of service/resource requests are to be fulfilled.

Secondary aspects to be also taken into consideration

Service/Resource allocation mechanisms. We need to simulate at least the economic centralised and decentralised service/resources allocation approaches shown in Figure1.1. Therefore, code implementing Catallactic and economy-based centralised mechanisms should be easily pluggable into the simulation framework.

Output for mechanism evaluation. We need to collect from simulation appropriate statistics and parameter values so that they can elaborated to drive conclusions about the quality of the allocation mechanism being evaluated.

In the next sections we describe the functional requirements for the simulation environment elicitated by analysing the four areas above.

2.1 ALN modelling

A reference model for a ALN should capture features that are relevant for ALN characterisation from both static and dynamic points of view. From a static point of view we see three main components which populate ALN nodes. They are:

Clients. Clients want to use the ALN in order to run their applications and satisfy their requests.

Services. Services are the components in the ALN whose goal is to satisfy clients requests. There can be several types of available services. For example there can be services of type *pdf converter*, or *search engine*, or *flight reservation*. For example, pdf converters can differ in relation of conversion quality or conversion time. Services can be atomic or composed by other services. For example a service of type *Travel Agency* might be composed by a service of type *train reservation* and a service of type *hotel booking*.

Resources. Services are not able by their own to accomplish requests from clients but need to exploit ALN resources. There can be several types of resources, typical examples are *CPU*, *bandwidth*, or *storage*. Resource of the same type can differ. For example, there can be CPUs having various computational powers or storage with different capacity.

From a dynamic point of view the ALN model should capture:

Service distribution. Services in the network might be highly distributed among nodes or concentrated in few nodes.

Resource distribution. Resources in the network might be highly distributed among nodes or concentrated in few nodes.

Configuration dynamism. The set of sites of a ALN and their interconnection can vary over time. Moreover, the distribution over the ALN nodes of resources and services can vary over time.

Network cost/delay. Accessing remote services or resources can have a cost, or service or resource provision from remote ALN sites may take longer then from close sites.

Usage patterns. In general, clients may request the same services recurrently or request different services each time. This means that we need a way to model different patters for service requests from ALN clients.

Given the characterisation above, we propose the following ALN model.

2.1.1 ALN model

An ALN is defined by a fully connected graph $< S, L >$ where S is a set of *sites* and L is a set of *network links* which connect sites. Each link $< s_i, s_j >$, where s_i and s_j denote ALN sites, is characterised by a delay coefficient $\delta_{<s_i,s_j>}$. The value of a delay coefficient is inversely proportional to the bandwidth between two sites and reflect how fast is the access to services or resources from pairs of ALN nodes.

Each site s is defined by a triple $< CSP, BSP, RP >$, where CSP is a set of *Complex Service Providers*, BSP is a set of *Basic Service Providers*, and RP is a set of *Resource Providers*. In every site there can be zero or more complex/basic service providers and zero or more resource providers, that is $|CSP| \geq 0, |BSP| \geq 0, |RP| \geq 0$.

Complex service providers offer *Complex Services (CSs)* to ALN *clients*. A complex service need a *set* of *Basic Services (BSs)* for fulfilling its goals. There are no assumptions on the ordering relationships between BSs in the set. In other words, we do not consider the notion of *workflow* in the specification of complex services. Complex service providers are not specialised. All of them are potentially able to provide clients with arbitrary sets of basic services. This means that a client will ask a complex service provides for a service according to her/his preferences, for example locality.

Basic service provides provide complex service providers with BSs they need to furnish their complex services to clients. Basic services may differ in type. For example there can be basic services of type *pdf converter*, or *search engine*, or *flight reservation*. If n basic service types are present in the market, then they are denoted by $bs_1, bs_2, \ldots, bs_n$.

There can be multiple instance of basic service providers that provide the same basic service type. A basic service provider bsp is characterised by the basic service type bs_i it provides. Multiple providers for the same basic service can co-exist in the market. They are indistinguishable from the point of view of quality of service.

Resource providers furnish basic service providers with resources they need to, in turn, furnish their basic services to complex service providers. Resources may differ in type, for example there can be resource, or *storage*, or *cpu*. If m resource types are present in the ALN, then they are denoted by $rt_1, rt_2, \ldots, rt_m$. A resource provider rp is characterised by the resource type rt_j it provides and a power p that defines the work it can perform in a time unit. This means that multiple instances of providers for the same resource but with different power can co-exist in the ALN. For example, if the resource type is *cpu*, then its power refers to computational power while if the resource type is *storage*, power refers to data access rate.

A basic service is specified by the *set* of resource requests which are needed so that the basic service can be provided. Each resource is requested to execute an amount of *work*. A resource request is a pair (r, w) where r denotes a resource type and w the workload the resource is requested to do. For example, if r denotes a *cpu*, then w represents a computational workload , while if it denotes a *storage*, w is the amount of data that must be read/write from/to the device. Sets of resource requests represent *bundles* of resources, i.e., resources which are needed as a whole. This implies that a resource allocation mechanism should be able to provide basic services with all the specified resources.

2.2 Service and resource allocation mechanisms

The key idea proposed by the CATNETS project is to adopt service and resource allocation mechanism based on the concept of *market*. Services and resources are seen as goods that are traded by clients, service providers and resource providers. The efficient allocation of services to clients and resources to services is supposed to be an emergent property of the market. The market model adopted in CATNETS is presented in details in Deliverable D1.1 [WP105]. Here, we just summarise its main features.

2.2.1 Service and resource markets

We assume there are two markets. The first is called *service market* and is the place where complex service provides and basic service providers trade for basic services. Clients are not modelled explicitly in the service market because they do not trade for (complex) services but just ask complex service providers according to their preferences[1]. The second market is called *resource market* and is the place where basic service providers and resource providers trade for resources. This reference scenario is illustrated in Figure 2.1.

In the *service market* we can identify the following *roles*:

Complex Service Provider. Clients need their jobs to be executed by purchasing services available on the service market. Complex Service Providers acts on behalf of one ore more clients and provide them with "best" services that are needed to execute their jobs. In order to do so, they negotiate with basic service providers.

Basic Service Provider. Its goal is to sell a service to Complex service Providers by maximising its incomes. It need translates abstract service demand to concrete technical resources, to be purchased in the resource market.

In the *resource market* we can identify the following *roles*:

---

[1]Note that, as previously stated, complex service providers are indistinguishable and therefore there is no competition among them.

Service Market   **Resource Market**

Complex Service Provider   Basic Service Provider   Resource Provider
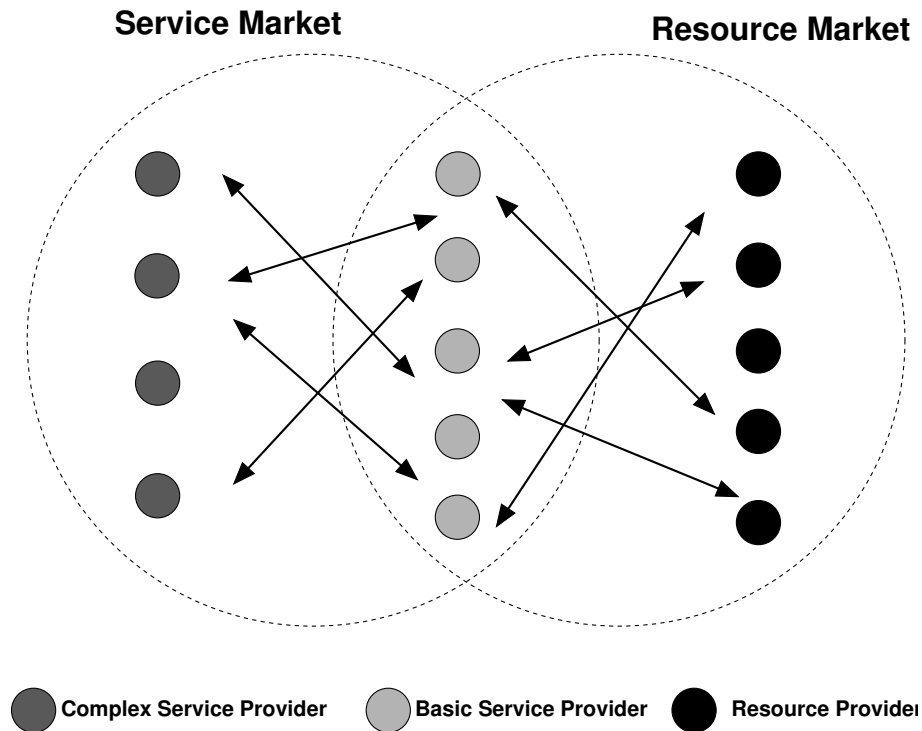
Figure 2.1: Service and resource markets.

Basic Service Provider. Each basic service needs a set of technical resources. A basic service provider needs to optimally purchase these resources on the resource market. In order to do so, it negotiates with resource providers. Technical resources needed for a basic service are to be purchased as a bunch. This has be taken into consideration in the definition of bargaining strategies used by basic service providers.

Resource Provider. Its goal is to sell a resource to basic service providers by maximising its incomes.

The simulation of two separate markets implies that it is not possible for a complex service to access a local resource manager directly. A complex service can only demand a basic service entity on the service market. A basic service entity translates service requests to resource demands on the resource market.

Service and resource selection in the two markets can be centralised or decentralised. In the centralised approach, demands and offers are matched by an auctioneer whose task is to periodically clear the market by matching offers from resource/service providers with requests (see Deliverable D1.1 [WP105] for additional details). The auctioneer act acts on behalf of all the agents in the market with the goal of maximising their utility. In the decentralised (Catallactic) scenario, each agent acts autonomously.

Depending on which allocation mechanisms is to be simulated, the simulator environment should be able to embed the the appropriate economic algorithms. In particular, for the Catallactic mechanism:

- a bargaining strategy which includes offer and price generation, selection of offers to negotiate with, learning, and recording a negotiation history;

- a protocol used for negotiation among market's players in either the service and resource market. As the approach is distribute, the most natural choice is to base negotiation protocol on message passing. Players in the market should be able to exchange messages related to call for bids, bids, acceptance or refusal of prices. This need to be easily modelled in the simulation environment. Moreover the simulation environment should also support multi-round negotiations.

As stated in the CATNETS Deliverable D1.1 [WP105], for the centralised economic approach the use of auctions in the two markets is an efficient way for the allocation of services and resource, as well as to determine their prices. The centralised auctioneer will be developed by WP1 and the way how it can be plugged into the simulator has to be jointly decided with them.

2.3   Scalability

There are several definitions of scalability of a computer system. A generally accepted definition is

> *How well a solution to some problem will work when the size of the problem increases.*

For application layer network, the size of the problem refers to dimension of the ALN to be simulated, the solution to the problem maps into the execution of the simulation. According to the ALN model we presented in Section 2.1, there are three parameter that contribute to the size of the ALN. They are:

1. the number of ALN nodes;

2. the total number of agents (Complex Service Providers, Basic Service Providers, Resource Providers) which operate in the ALN;

3. the number of service and resource requests that are to be satisfied.

These parameters are independent each other since, for example, there can be ALNs with a small number of nodes but where the density of agents is elevated. Moreover, The rate

of complex/basic service or resource requested per agent is clearly independent from the number of agents or nodes in the ALN.

A simulator whose aim is to evaluate the performance of a service/resource allocation mechanism in a simulated but realistic ALN should be able to run in reasonable time scenarios where the values of the parameters above are realistic. As stated in Deliverable D3.1 [], the concept of ALN maps essentially into three types of applications: Content Distributed Networks, Peer-to-Peer Networks, and Grids. From these prototypical applications we can derive the following estimations for the three parameters that defines the size of a ALN.

Number of ALN nodes. The value of this parameter ranges from few tens to 2-3 hundreds.

Number of agents. The value of this parameter ranges from about 100 to to 5-6 hundreds.

Number of service and resource requests. The value of this parameter is in the order of several thousands.

The estimation of the number of service and resource requests has been done by considering to factors: the request rate typical of the considered ALN applications and the need of being able to simulate the ALN behaviour for a sufficiently long time. This to avoid perturbations due to the initial transitory.

2.4   Output for mechanism evaluation

The output of the simulation need to be used for the evaluation of the service/resource allocation mechanism. Economical and technical evaluation metrics are defined by WP4-Evalutaion and presented in Deliverable D4.1 [WP405]. In summary, the simulator should be able to output traces related to the technical metrics that will be used by an external evaluation module to build up the economic metrics. In order to make simulation independent from evaluation, the chosen format for these traces is a file. The schema of the output data will be decided in agreement with WP4-Evaluation.

Chapter 3

Simulators

The massive increase of network application has generated a lot of interest and effort in understanding the way to make them more efficient. For this reason, an increasing number of Grid, P2P, internet and network simulators is available, each of them stressing particular simuation aspects. The work by Sulistio et al. [SYB04] gives an informed vision of such a trend in the research community. From that work we extract Figures 3.1, that provides us with an overview of a number of existing simulators, and Figure 3.2, that gives some insights into each of them.

It is quite clear that a comprehensive evaluation of the gret number of existing simulators is very difficult to achieve. Moreover, by rely on assessments done by others we have the problem that the requirements chosen by evaluation do not match with ours. For these reasons we decided to assess simulators over the requirements presented in Chapter 2 and restrict evaluation over a set of simulators that we know. In the following we give description for such simulators. Section 3.1 presents simulators direclty developed or extensively used by consortium members, while Section 3.2 includes description of simulators on which we have less deep knowledge but we think could be reasonable candidates for CATNETS.

3.1 The consortium experience

3.1.1 Catnet simulator

In the CATNET assessment project a simple simulator was developed to give a preliminary and partial evaluation of the behavior of a P2P system using a Catallactic coordination mechanism. CATNET is a simulator for an application layer network, which allows creating different types of agents to form a network. This simulator is implemented on

Category	Tool	Organization	Key similarities and differences, simulated systems, and Web site
Parallel systems	SimOS	Stanford University, U.S.A.	• Models complete computer systems through fast simulation of hardware and levels of abstraction. • Simulates a complete multiprocessor system and studies all various aspects including hardware architecture, operating system and application programs. • http://simos.stanford.edu/
Distributed systems	SimJava	University of Edinburgh, U.K.	• Provides a core set of foundation classes for simulating discrete events. • Simulates distributed hardware systems, communication protocols and computer architectures. • http://www.dcs.ed.ac.uk/home/simjava/
Networks	NS-2	University of California at Berkeley, U.S.A.	• Supports several levels of abstraction to simulate a wide range of network protocols via numerous simulation interfaces, such as using scripting language and/or system language. • Simulates network protocols over wired and wireless networks. • http://www.isi.edu/nsnam/ns/
	Parsec	University of California at Los Angeles, U.S.A.	• Uses a portable runtime kernel that executes simulations on either sequential or parallel architectures enhanced by ready support of numerous parallel simulation protocols. • Simulates very large scale integrated (VLSI) parallel architectures, parallel databases and wireless networks using parallel simulation. • http://pcl.cs.ucla.edu/projects/parsec/
Mobile systems	GloMoSim	University of California at Los Angeles, U.S.A.	• Provides an extensible and modular library that supports implementation of alternative protocols for each layer of the wireless communication protocol stack. • Simulates large-scale wireless mobile networks. • http://pcl.cs.ucla.edu/projects/glomosim/
Grid scheduling systems	Bricks	Tokyo Institute of Technology, Japan	• Provides simulation for resource allocation strategies and policies for multiple clients and servers as in global computing systems in a Grid environment. • Simulates resource scheduling algorithms in Grids. • http://matsu-www.is.titech.ac.jp/~takefusa/bricks/
	GridSim	University of Melbourne, Australia	• Supports simulation of space-based and time-based, large-scale resources in the Grid environment. • Simulates economy-based resource scheduling systems in Grids. • http://www.gridbus.org/gridsim/
	MicroGrid	University of California at San Diego, U.S.A.	• Runs emulations by executing actual application code on the virtual Globus Grid and thus requires more time to complete the application. • Emulates the Globus Grid environment for resource management. • http://www-csag.ucsd.edu/projects/grid/
	SimGrid	University of California at San Diego, U.S.A.	• Simulates a single or multiple scheduling entities and time-shared systems operating in a Grid computing environment. • Simulates distributed Grid applications for resource scheduling. • http://grail.sdsc.edu/projects/simgrid/
Embedded systems	Ptolemy II	University of California at Berkeley, U.S.A.	• Builds upon a component-based design methodology that hierarchically integrates multiple models of computation to capture different design perspectives. • Simulates systems that comprise heterogeneous components and sub-components. • http://ptolemy.eecs.berkeley.edu/ptolemyII/

Figure 3.1: A wide list of simulators from [SYB04].

Design	SimOS	SimJava	NS-2	Parsec	GloMoSim
Simulated systems	Parallel systems	Distributed systems, networks	Wired and wireless networks	VLSI circuits, wireless networks, parallel architectures	Wireless mobile networks
Usage	Simulator	Simulator	Simulator	Simulator	Simulator
Simulation	Static, discrete, deterministic	Static, discrete, deterministic	Static, discrete, deterministic	Static, discrete, deterministic	Static, discrete, deterministic
Simulation engine	Parallel, event-driven DES	Serial, event-driven DES	Serial, event-driven DES	Serial & parallel, event-driven DES	Serial & parallel, event-driven DES
Modeling framework	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based
Programming framework	Structured	Object-oriented	Object-oriented	Structured	Structured
Design environment	Language	Library	Language	Language, library	Library
User interface	Non-visual	Animation, graph	Animation	Drag-drop, form	Non-visual
System support	Debugging, statistics generation	Statistics generation	Debugging, statistics generation, validation test	Code generation	Statistics generation

Design	Bricks	GridSim	MicroGrid	SimGrid	Ptolemy II
Simulated systems	Grid, resource scheduling systems	Grid, resource scheduling systems	Grid, resource scheduling systems	Grid, resource scheduling systems	Embedded systems
Usage	Simulator	Simulator	Emulator	Simulator	Simulator
Simulation	Static, discrete, deterministic	Static, discrete, deterministic	Dynamic, continuous, deterministic	Static, discrete, deterministic	Dynamic, continuous, deterministic
Simulation engine	Serial, event-driven DES	Multithreaded, event-driven DES	Parallel, event-driven DES	Serial, trace-driven DES	Serial, hybrid
Modeling framework	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based	Entity-based, event-based
Programming framework	Object-oriented	Object-oriented	Structured	Structured	Object-oriented
Design environment	Language	Library	Language	Library	Language, library
User interface	Non-visual	Form	Non-visual	Non-visual	Drag-drop, form, graph
System support	Statistics generation	Code generation, statistics generation	N/A	N/A	Code generation, debugging, statistics generation

Figure 3.2: Features of the simulators from [SYB04].

top of the J-Sim network simulator.[1]

The CATNET simulator implements two main control mechanisms for the network coordination: the baseline and the Catallactic control mechanism. The baseline mechanism computes the service/resource allocation decision in a centralized way. In the Catallactic mechanism, autonomous agents take their decisions in a decentralized way, having only local information about the environment. Each agent disposes of a strategy to take decisions, which targets to increase the agents own benefit. In the simulations, a service is considered as the functionality, which is exchanged among the peers in the network. The concept of service and the functions, or "personalities", a peer can assume in the CATNET simulator, are the following:

Service. A service encapsulates a general function performed in the P2P network. A service is the provision of a resource such as computing power, data storage, content, or bandwidth. The service provision includes the search for a resource and its reservation for availability.

Client. A peer may act as a client or consumer of a service. As such it needs to access the service, use it for a defined time period, and then continues with its own program sequence.

Resource. A peer, which is the owner of a required functionality. This functionality, for instance, may represent content, storage or processing power. The functionality, which is required by the clients or consuming peers, is encapsulated in a service.

Service copy. A peer acting as a service copy offers a service as an intermediary, however it is not the owner of the components to provide the service. It must cooperate with the resource to be able to provide the service. Service copies offer the service to requesting clients.

In the simulator, the application layer network is built on top of a physical network topology. The physical network topology is specified in the input of the simulator. The topology could be random or having a determined structure specified by the user. A node can host several agents or none at all. In the latter case, the node just acts as a router.

During the inizialization process several features are set:

- the capacity of resources;

- the initial prices of Clients, Service Copies, and Resource agents;

- initial budget of Clients;

---

[1]J-Sim simulates a general TCP/IP network and provides substantial support for simulating real network topologies and application layer services, i.e. data and control messages among application network instances.

- the type of control mechanism (baseline or Catallactic).

The CATNET simulator allows to vary two important parameters of the application layer network:

1. node dynamics;

2. node density.

Node dynamics measures the degree of availability of service-providing nodes in the network. Low dynamics mean an unchanging and constant availability; high dynamics are attributed to a network where nodes start up and shut down with great frequency. Node density measures the relation of resource nodes to the total number of network nodes. The highest density occurs when every network node provides the described service to others; the lowest density is reached if only one resource node in the whole network exists.

The CATNET simulator was employed to compare the two control mechanisms conducting for each of them 9 simulations corresponding to three different levels of node dynamics (null, medium and high) and three levels of node density (low, medium and high)[2]. The following table reports the description of a typical experiment

Input trace	- 2000 service requests generated randomly by 75 clients over a time interval of 100 s. - each request is for 2 service units. - each service has a duration of 5 s.
Node topol.	- 106 physical nodes
Node density	- 75 clients on the leaves of the physical network - different density of resource and service copy agents. Each Resource has one service copy associated. Exp 1A: low node density: 5 resources with capacity 60. Exp 1B: medium node density: 25 resources with capacity 12. Exp 1C: high node density: 75 resources with capacity 4.
Node dynamics	Dynamic behavior: On average 70% of the service copies are connected. Exp 2A: Service copies do not change its state (static network) Exp 2B: Each 200 ms every service copy can change its state (connected/disconnected) with a probability of 0.2. Exp 2C: Each 200 ms every service copy can change its state (connected/disconnected) with a probability of 0.4.

Table 3.1: Results of a typical experiment done with the CATNET simulator.

3.1.2 OptorSim

OptorSim ([CCSM⁺04, BCC⁺03, opt]) is a joint effort of ITC-irst, University of Glasgow and CERN. It is a open-source Data Grid simulator that has been developed in the

---

[2]18 basic experiments was conducted in total.

framework of the European DataGrid (EDG) [edg] in order to explore by simulation the behaviour of different data replication algorithms in several Grid scenarios, in particular related to the field of High Energy Physics.

Simulation Design

There are a number of elements which should be included in a Grid simulation to achieve a realistic environment. These include: computing resources to which jobs can be sent; storage resources where data can be kept; a scheduler to decide where jobs should be sent; and the network which connects the sites. For a Grid with automated file replication, there must also be a component to perform the replica management. It should be easy to investigate different algorithms for both scheduling and replication and to input different topologies and workloads.

Architecture

OptorSim is designed to fulfil the above requirements, with an architecture (Figure 3.3) based on that of the EDG data management components. In the model, computing and
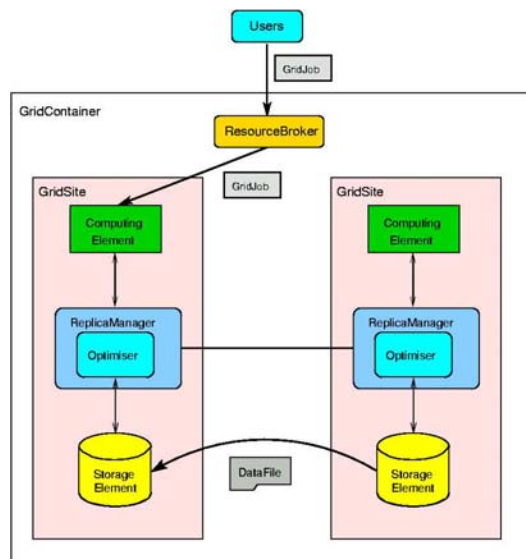

Figure 3.3: OptorSim Architecture.

storage resources are represented by *Computing Elements* (CEs) and *Storage Elements* (SEs) respectively, which are organised in *Grid Sites*. CEs run jobs by processing data files, which are stored in the SEs. A *Resource Broker* (RB) controls the scheduling of jobs to Grid Sites. Each site handles its file content with a *Replica Manager* (RM), within

which a *Replica Optimiser* (RO) contains the replication algorithm which drives automatic creation and deletion of replicas.

Input Parameters

A simulation is set up by means of configuration files: one which defines the Grid topology and resources, one the jobs and their associated files, and one the simulation parameters and algorithms to use. The most important parameters include: the access pattern with which the jobs access files; the submission pattern with which the users send jobs to the RB; the level and variability of non-Grid traffic present; and the optimisation algorithms to use. A full description of each is in the OptorSim User Guide [BCCS⁺04].

Optimisation Algorithms

There are two types of optimisation which may be investigated with OptorSim: the scheduling algorithms used by the RB to allocate jobs, and the replication algorithms used by the RM at each site to decide when and how to replicate.

Scheduling Algorithms. The job scheduling algorithms are based on reducing the "cost" needed to run a job at a particular site. The algorithms currently implemented in OptorSim are: *Random* (a site is chosen at random); *Access Cost* (cost is the time needed to access all the files needed for the job); *Queue Size* (cost is the number of jobs in the queue at that site); and *Queue Access Cost* (the combined access cost for every job in the queue, plus the current job).

Replication Algorithms. There are three broad options for replication strategies in OptorSim. Firstly, one can choose to perform no replication. Secondly, one can use a "traditional" algorithm which, when presented with a file request, always tries to replicate and, if necessary, deletes existing files to do so. Algorithms in this category are the LRU (Least Recently Used), which deletes those files which have been used least recently, and the LFU (Least Frequently Used), which deletes those which have been used least frequently in the recent past. Thirdly, one can use an economic model in which sites "buy" and "sell" files using an auction mechanism, and will only delete files if they are less valuable than the new file. Details of the auction mechanism and file value prediction algorithms can be found in [BCCS⁺03]. There are currently two versions of the economic model: the binomial economic model, where file values are predicted by ordering the files in a binomial distribution about the mean file index in the recent past δT, and the Zipf economic model, where the values are calculated by ordering them in a Zipf-like distribution according to their popularity in δT.

Implementation

OptorSim is a time-based simulation package written in Java. Each CE is represented by a thread, with another thread acting as the RB. There are two time models implemented. In *SimpleGridTime*, the simulation proceeds in real time. *AdvancedGridTime*, on the other hand, is semi-event driven; when all the CE and RB threads are inactive, simulation time is advanced to the point when the next thread should be activated. The use of *Advanced-GridTime* speeds up the running of the simulation considerably, whereas *SimpleGridTime* may be desirable for demonstration or other purposes.
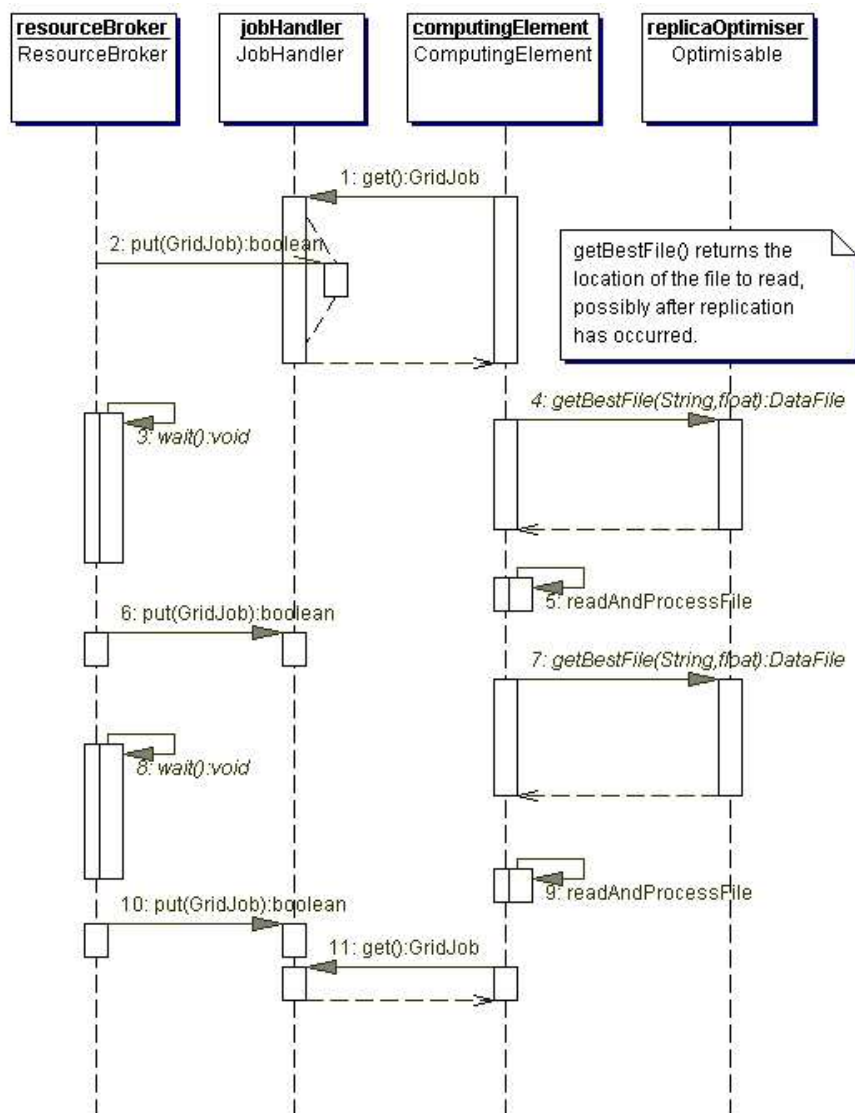
Figure 3.4: Sequence diagram of the Resource Broker and Computing Element threads.

A sequence diagram of some of the run-time interactions is shown in Figure 3.4. The RB sends jobs to the CEs according to the specified scheduling algorithm and the CEs process the jobs by accessing the required files, running one job at a time. In the current implementation, the number of worker nodes for each CE simply reduces the time a file takes for processing, rather than allowing jobs to run simultaneously. When a file is needed, the CE calls the `getBestFile()` method of the RO being used. The replication algorithm is then used to search for the "best" replica to use. Each scheduling and replication algorithm is implemented as a separate Resource Broker or Replica Optimiser class respectively and the appropriate class is instantiated at run-time, making the code easily extensible.

OptorSim can be run from the command-line or using a graphical user interface (GUI). A number of statistics are gathered as the simulation runs, including total and individual job times, number of replications, local and remote file accesses, volume of storage filled and percentage of time that CEs are active. If using the command-line, these are output at the end of the simulation in a hierarchical way for the whole Grid, individual sites and site components. If the GUI is used, these can also be monitored in real time.

Experimental Setup

Two Grid configurations which have been simulated recently are the CMS[3] Data Challenge 2002 testbed (Figure 3.5) and the LCG[4] August 2004 testbed (Figure 3.6).

For the CMS testbed, CERN and FNAL were given SEs of 100 GB capacity and no CEs. All master files were stored at one of these sites. Every other site was given 50 GB of storage and a CE with one worker node. For the LCG testbed, resources were based on those published by the LCG Grid Deployment Board for Quarter 4 of 2004 [lcg], but with SE capacities reduced by a factor of 100 and number of worker nodes per CE halved. All master files were placed at CERN. In both cases, the dataset size was 97 GB.

Testbed	No. of Sites	$D/\langle SE \rangle$	$\langle WN \rangle$	$\langle C \rangle$ (Mbit/s)
CMS	20	1.764	1	507
LCG	65	0.238	108	463

Table 3.2: Comparison of Testbeds Used.

In order to compare results from these testbeds, it is necessary to summarise their main characteristics. Useful metrics are: the ratio of the dataset size to the average SE size, $D/\langle SE \rangle$; the average number of worker nodes per CE, $\langle WN \rangle$; and the average

---

[3]Compact Muon Solenoid, one of the experiments for the Large Hadron Collider (LHC) at CERN.

[4]A project whose mission of the is to build and maintain a data storage and analysis infrastructure for the entire high energy physics community that will use the LHC.

Figure 3.5: CMS Data Challenge 2002 Grid topology.

Figure 3.6: LCG August 2004 Grid topology.
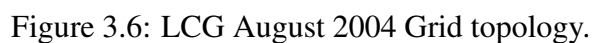
connectivity of a site, $\langle C \rangle$. The values of these metrics for the two testbeds are shown in Table 3.2. Some general statements can be made about these characteristics:

- $D/\langle SE \rangle$. A low value of $D/\langle SE \rangle$ indicates that the SEs have more space than is required by the files. Little deletion will take place and one would expect the different replication algorithms to have little effect.

- $\langle WN \rangle$. A high value of $\langle WN \rangle$ will result in jobs being processed very quickly. If the job processing rate is higher than the submission rate, there will then be little queueing and the mean job time will be short. A low number of worker nodes could lead to processing rate being lower than the submission rate and thus to escalating queues and job times.

- $\langle C \rangle$. A high $\langle C \rangle$ will result in fast file transfer times and hence fast job times. This will have a similar effect on the ratio of job processing rate to submission rate as described above for $\langle WN \rangle$.

Another important factor is the presence or absence of a CE at the site(s) which initially hold(s) all the files. In OptorSim, the intra-site bandwidth is assumed to be infinite, so if a file is local there are no transfer costs involved. For scheduling algorithms which consider the transfer costs, most of the jobs will therefore get sent to that site.

Results

CMS Data Challenge 2002 testbed. First, three of the replication algorithms (LFU, binomial economic and Zipf-based economic) were compared for the four scheduling algorithms, with 1000 jobs on the Grid. The mean job times are shown in Figure 3.7. This shows that scheduling algorithms which consider the processing cost of jobs at a site possess a clear advantage, as mean job time is reduced considerably for the *Access Cost* and *Queue Access Cost* schedulers. It can also be seen that the LFU replication algorithm is faster than the economic models for this number of jobs. This may be due to the low value of $\langle WN \rangle$; as the economic models have an overhead due to the auctioning time, there will initially be more queue build-up than with the LFU.

A study was also made of how the replication algorithms reacted to increasing the total number of jobs (Figure 3.8). As the number of jobs on the Grid increases, the mean job time also increases. One would expect that it should decrease if the replication algorithms are effective, but with the low value of $\langle WN \rangle$ in this case, the job submission rate is higher than the processing rate, leading to runaway job times. However, the performance of the economic models improves in comparison to the LFU and when 10,000 jobs are run, the Zipf economic model is faster. For long-term optimisation, therefore, the economic models could be better at placing replicas where they will be needed.
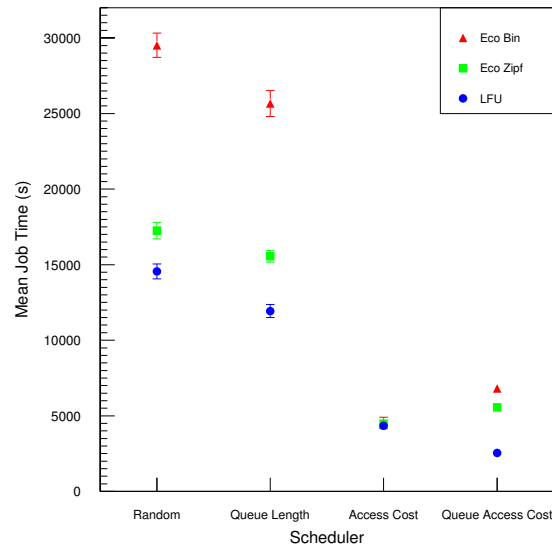
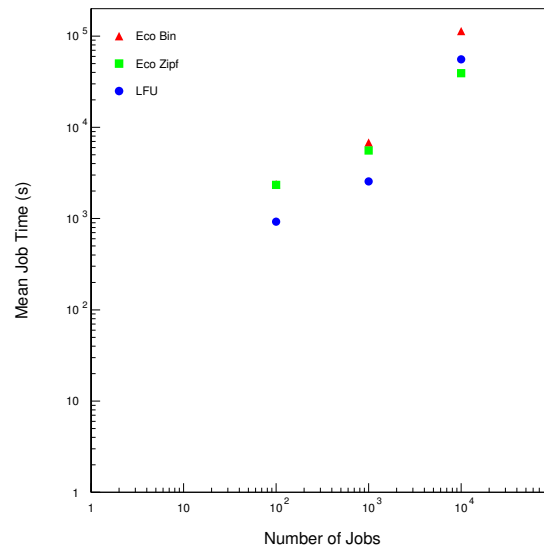Figure 3.7: Mean job time for scheduling and replication algorithms in CMS 2002 testbed.

Figure 3.8: Mean job time for increasing number of jobs in CMS 2002 testbed.

LCG August 2004 testbed. The pattern of results for the scheduling algorithms (Figure 3.9) are similar to those for the previous configuration. The *Access Cost* and *Queue Access Cost* algorithms are in this case indistinguishable, and the mean job time for the LFU algorithm is negligibly small. This is due to the fact that in this case, CERN (which contains all the master files) has a CE. When a scheduler is considering access costs, CERN will have the lowest cost and the job will be sent there. This is also a Grid where
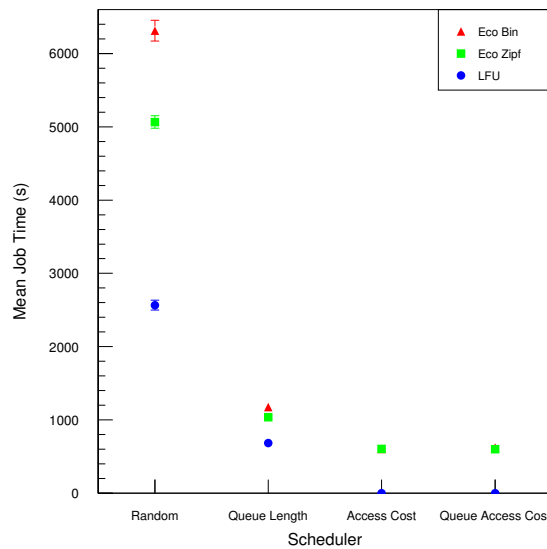
Figure 3.9: Mean job time for scheduling and replication algorithms in LCG August 2004 testbed.

the storage resources are such that a file deletion algorithm is unnecessary and a simple algorithm such as the LFU runs faster than the economic models, which are slowed down by the auctioning time. It would therefore be useful to repeat these experiments with a heavier workload, such that $D/\langle SE \rangle$ is large enough to reveal the true performance of the algorithms.

3.1.3 Agent Based simulators

UPM (Università Politecnica delle Marche) has experience in developing economic models using the agent based simulators described in this section.

SWARM

SWARM [swa] was originally a written in Objective C but currently it comprises a set of libraries which allows simulations of agent based models written either in Objective C or

Java. These libraries work on a very wide range of computer platforms.

In SWARM a simulation is organised in two levels: the first level specifies the interactions between the user and the simulator (called the observer) while the second the model of the scenario to be simulated. Both the observer and the model are developed in two phases. In the first phase the agents are defined by specifying the actions they can perform and then created. In the second phase the dynamics of the model is designed by giving to each agent its own schedule.

The possibility to interact with the model is particularly useful in the preliminary stage of simulation when the behavior of the model has to be checked to discover possible mistakes. The presence of a GUI make the simulation rather slow and it should be avoided in the second phase of simulation (when one tries to increase the size of the simulation).

SWARM has a wide user community. Is ha been used for:

Biological simulation: Bacterial Growth, Ecosystem Dynamics, Nerual Networks, Bee Swarm behaviour, Metabolizing Agents.

Ecology: Animal migration, Plant ecosystem evolution.

Computing: Analysis of load balancing on multiple processors, Analysis of multi-agent manufacturing, computer network analysis.

Economics: Stock Market simulation.

Political Science: Political Party formation simulation.

Geography: Traffic pattern analysis, Land use analysis.

Military: Weapon Deployment analysis.

RePast

The *Recursive Porous Agent Simulation Toolkit (Repast)* [rep] is an open source toolkit that was originally developed by Sallach, Collier, Howe, North and others at the University of Chicago [CHN03]. Repast borrows many concepts from the SWARM toolkit. Repast differentiates from SWARM since Repast has multiple pure implementations in several languages and built-in adaptive features such as genetic algorithms and regression. For reviews of SWARM, Repast, and other agent-modeling toolkits, see the survey by Serenko and Detlor, the survey by Gilbert and Bankes, and the toolkit review by Tobias and Hofmann [SD02, GB02, TH03].

Repast can be thought of as a specification toolkit for agent-based services or functions. There are three concrete implementations of this conceptual specification. All of these versions have the same core services that constitute the Repast system. The implementations differ in their underlying platform and model development languages. The

three implementations are Repast for Java (Repast J), Repast for the Microsoft.Net framework (Repast.Net), and Repast for Python Scripting (Repast Py). Repast J is the reference implementation that defines the core services. In general, it is recommended that basic models can be written in Python using Repast Py due to its visual interface and that advanced models be written in Java with Repast J or in C# with Repast .Net.

Repast has a variety of features including:

- users have complete flexibility as to how they specify the properties and behaviors of agents;

- full object-orientation;

- fully concurrent discrete event scheduler which supports both sequential and parallel discrete event operations;

- built-in simulation results logging and graphing tools;

- automated Monte Carlo simulation framework;

- range of two-dimensional agent environments and visualisations;

- users can dynamically access and modify agent properties, agent behavioral equations, and model properties at run time;

- libraries for genetic algorithms, neural networks, random number generation, and specialized mathematics;

- built-in systems dynamics modeling;

- social network modeling support tools;

- Repast has integrated geographical information systems (GIS) support.

Repast is available on virtually all modern computing platforms. The platform support includes both personal computers and large-scale scientific computing clusters.

JAS

JAS (Java Agent-based Simulation library) [jas] is a Java toolkit for creating agent-based simulations. It features a discrete-event time engine, statistical probes with Hypersonic database built-in storage capability, Neural Networks and Genetic Algorithms packages, graph support for Social Network Analysis. JAS have some useful features:

- discrete-event time simulation engine;

- management of several time units (ticks, seconds, minutes, days...);

- the real time engine is able to fire events using the real computer timer;

- support for XML data I/O and SVG file format;

- genetic algorithms, neural networks, (classifier systems, still under construction);

- Sim2Web, a JAS-Zope bridge for web publishing of simulations and remote users interaction.

- multirun support for automatic parameters calibration.

- statistical package with file and database I/O features.

3.2 Other simulators

In the previous section the description of the consortium direct experience in simulation was given. In this section we concentrate on simulators that, according to our indirect experience, could be suitable for CATNETS.

3.2.1 P2Psim

P2psim is a freeware, multi-threaded, discrete event simulator to evaluate, investigate, and explore peer-to-peer protocols. p2psim runs in Linux and FreeBSD and is part of the IRIS project. The goals of this simulators are:

1. to make understanding peer-to-peer protocol source code easy;

2. to make comparing different protocols convenient;

3. to have reasonable performance.

P2psim supports several peer-to-peer protocols, making comparisons between different protocols convenient. P2psim maximizes concurrency for performance, minimizes the need for synchronization, and avoids deadlocks.

3.2.2 PlanetSim

PlanetSim is an object oriented simulation framework for overlay networks and services. It has a layered and modular architecture with well defined hotspots documented using classical design patterns. In PlanetSim developers can work at two main levels: creating and testing new overlay algorithms or creating and testing new services on top of existing overlays.

PlanetSim also aims to enable a smooth transition from simulation code to experimentation code running in the Internet. A wrapper code that takes care of network communication and permits to run the same code in network testbeds such as PlanetLab is provided. Moreover, distributed services in the simulator use the Common API for Structured Overlays. This enables complete transparency to services running either against the simulator or the network.

PlanetSim has been developed in Java and is optimised to enable scalable simulations in reasonable time.

3.2.3 Peersim

Peer-to-peer systems can reach notable dimension and nodes typically join and leave continuously. Thus, for dynamic large-scale systems, a scalable simulation testbed is mandatory.

Peersim has been developed with high scalability and support for dynamicity in mind. It is released under the GPL open source licence. It is composed of many simple extendable and pluggable components, with a flexible configuration mechanism. To allow for scalability and focus on self-organization properties of large scale systems, some simplifying assumptions have been made, such as ignoring the details of the transport communication protocol stack. Peersim is developed within the BISON project and is written in Java.

3.2.4 Diet Agents

The DIET Agents open source platform was created as part of the DIET project, where DIET stands for Decentralised Information Ecosystem Technologies. This was a European collaboration project funded by the European union under the Framework 5 program. The DIET project was part of the Information Societies Technologies projects, more specifically the Universal Information Ecosystem Initiative. The aim of DIET was:

- To study, implement and validate a novel information processing and management framework via a "bottom up" and ecosystem-inspired approach leading to an open, robust, adaptive and scalable environment.

- To research into the effects of alternative forms of agent-interaction under an ecological model, using techniques from Evolutionary Computation and Artificial Life.

3.2.5 GridSim

The GridSim toolkit [SPBT05] allows modeling and simulation of entities in parallel and distributed computing systems-users, applications, resources, and resource brokers

(schedulers) for design and evaluation of scheduling algorithms. It provides a comprehensive facility for creating different classes of heterogeneous resources that can be aggregated using resource brokers for solving compute and data intensive applications. A resource can be a single processor or multi-processor with shared or distributed memory and managed by time or space shared schedulers. The processing nodes within a resource can be heterogeneous in terms of processing capability, configuration, and availability. The resource brokers use scheduling algorithms or policies for mapping jobs to resources to optimize system or user objectives depending on their goals.

Salient funtionalities of the GridSim toolkit include:

- Modelling of heterogeneous types of resources.

- Resources can be modeled operating under space or timeshared mode.

- Resource capability can be defined in the form of MIPS (Million Instructions Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmark .

- Resources can be located in any time zone.

- Weekends and holidays can be mapped depending on resource's local time to model nonGrid (local) workload.

- Resources can be booked for advance reservation.

- Applications with different parallel application models can be simulated.

- Application tasks can be heterogeneous and they can be CPU or I/O intensive.

- There is no limit on the number of application jobs that can be submitted to a resource.

- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be timeshared or spaceshared. This feature helps in building schedulers that can use different marketdriven economic models for selecting services competitively.

- Network speed between resources can be specified.

- It supports simulation of both static and dynamic schedulers.

- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim

Chapter 4

Evaluation

In this chapter we assess the simulators presented in Chapter 3 with respect to the requirements described in Chapter 2. We first give a qualitative evaluation of the simulators. The qualitative analysis allows the selection of two simulation frameworks for which we give a quantitative evaluation in the second part of the chapter.

4.1 Qualitative evaluation

Qualitative evaluation is based on a two dimensional space where dimensions are scalability and specificity. A simulator is considered as more specific as more it is able to potentially simulate ALNs having the features presented in Section 2.1.
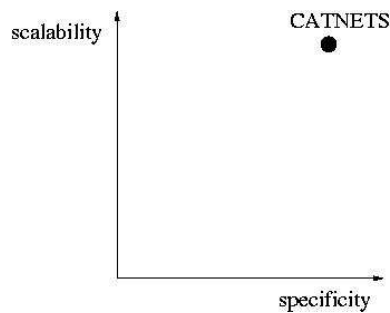

Figure 4.1: Scalability vs specificity

In the qualitative analysis we also take into account the two secondary aspects (service/resource allocation mechanisms and output for mechanism evaluation) presented in Chapter 2).

4.1.1 Specificity

Two frameworks seem to be tailored to naturally simulate an ALN corresponding to the model presented in Chapter 2. They are OptorSim, realised with the purpose of modelling Data Grids, and the CATNET simulator, developed in the assessment project with the specific goal of being a simulator for a ALN. Both simulators allow the specification of simulation scenarios where the building blocks maps quite well into the principal components that statically characterise the ALN model presented in Section 2.1.

All the other simulators described in Chapter 3 are not tailored for the rapid specification of an ALN having the described model because the building blocks for the definition of a simulation configuration are too low level. They are general purpose simulators which provide basic functionalities for the simulation of a wide range of application domains (including ALNs) but, by using them, we believe there is a risk that specification of the complex components of an ALN would take too long time. An exception is GridSim where basic entities of simulations corresponds to components of a computational or data grid.

Considering the ALN dynamic parameters presented in Section 2.1, OptorSim can model ALNs where there is resource and service distribution, network cost and usage patters, but are not dynamic. In addition, the CATNET simulator gives the uses the possibility of simulating configuration dynamism but does not permit the definition of specific distributions of ALN usage.

4.1.2 Scalability

Taking into account scalability, qualitative evaluation performed by examining simulators' documentation gives results which seems to be complementary with respect to the previous: general purpose simulator are usually more scalable than ALN-oriented simulators. In general, all general purpose simulator presented in Chapter 3 include mechanisms for increasing scalability. In the following we give insights for some of them.

- The agent based simulators presented in Section 3.1.3 allows run simulation with a great number of agents. The following sentence reported from the SWARM mailing list is significant: *We have done some some substantial size simulations of around a million agents distributed over 100 nodes of a cluster. Scaling was pretty impressive, with 40 times speedup achieved.* In summary, in these systems scalability is achieved by avoiding to simulate agents using threads but lightweight objects.

- In **Peersim** a time-stepped simulation model (instead of more complex and expensive event-based architecture) is adopted to increase scalability. At each time step, all nodes in the system are selected in a random order and, by means of the invocation of a callback method, all simulated entities running on that node get a chance to execute at each cycle.

- According to the website, using DIET it is possible to run over 100,000 agents on an ordinary desktop machine and there are no inherent limitations on scalability when running applications across multiple machines.

For ALN-oriented simulators, we have the following scalability information

- The current implementation of **OptorSim** allows the simulation of Data Grids having up to 70 sites. In each site there can be up to 4 agents modelling grid site's components. The number of requests for data file that can be dealt contemporary is around few hundreds. During a simulation, several thousands of file requests are satisfied.

- The CATNET simulator is able to run scenarios where ALN have up to about 100 sites. In the ALN there can be at most 200 services copies and resources. The number of service requests that can be satisfied is in the order of few thousands

4.1.3 Secondary aspects

Service and resource allocation mechanisms. The Catallactic distributed allocation mechanism is a novel concept and the simulator developed in the assessment project CATNETS include the only, even prototypical and not complete, implementation currently known to us. All the other simulators presented in the previous chapter do non embed anything similar. However, OptorSim embeds a tunable message flooding mechanisms for the negotiation of data files in the simulated grid. Such a mechanism is a good candidate as the base of the negotiation protocol needed for implementing Catallaxy.

As far as the centralised auction-based allocation mechanism is concerned, it will be developed as a separate software module to be plugged in into the simulator. Clearly, this is envisaged to be easier for simulators whose implementation is well known to us.

Output for mechanism evaluation. The detailed structure of the output files needed for the evaluation of the Catallactic and centralised allocation mechanisms are still to be discussed with WP4-Evaluation. For this reason it is not possible at the current time to assess the simulators with respect to this requirement.

4.1.4 Qualitative evaluation outcome

The choice of the level of specificity for the CATNETS simulator is directly driven by goals of the project: it is to analyse the efficiency of a system with respect to different service/resource allocation mechanisms (centralized vs decentralized) for ALNs. So we conclude that the simulator should be specific for this purpose but generic within the purpose: it should be able to easily include different allocation mechanisms.

From the requirements presented in Section 2.3 and related to scalability we can drive the conclusion that the CATNETS simulator should be highly scalable in terms of the number of service requests to be satisfied but it is not strictly necessary, even though preferably, that it is able to run simulation with a huge number of ALN nodes.

By taking into account also the non functional requirements illustrated at the beginning of Chapter 2, we see two possibilities for the CATNET simulator. The first is to use the an agent based simulator (Repast) that is used in disciplines like biology, economy, social systems and so on to simulate systems with a huge amount of agents. The drawback of this simulation framework is that it is are general purpose and must be specialized for CATNETS goals. The second one is OptorSim that already provides building blocks for a simulations that correspond very well to the principal components of ALNs and could be adapted with limited effort. However, t could have some problem for scalability. Both the framework should be integrated with software module for bargain strategies and negotiation, taken from the CATNET simulator.

In order to reasonably choose between Repast and OptorSim, we quantitatively evaluate them in the next section.

4.2 Repast vs OptorSim

Quantitative evaluation is based again on scalability and specificity and also on a set of features related to execution facilities provided by the two frameworks.

4.2.1 Method

To evaluate the two candidate simulators it is useful to build a numeric indicator. To calculate the value of the indicator we identify some relevant features and, for each feature, give to the simulator a score (s_i) that ranges from 1 to 5. As each features has different importance in the evaluation we assign to each feature a weight (w_i). Each weigh is a value in the range $(0, 1)$ and the sum of weight is 1. The final indicator is a weighted sum of the scores.

$$indicator = \sum_{i=1}^{7} s_i * w_i$$

4.2.2 Features

We define four features: *specificity*, *scalability*, *framework facilities*, and *user community*. Specificity and scalability are given weight 0.3, framework facilities weight 0.3, and user community weight 0.1. Actually, framework facilities is a set of 4 features concerning

different aspect of the simulators, as detailed in the following. Each of these features is given weight 0.075.

Specificity. The CATNETS simulator need to be realised with reasonable effort and avoiding as much as possible the introduction of mistakes in the developed code. For this reasons, specificity is an important feature to be taken into account. Specificity is concerned with two aspects:

> **New code:** inverse of the effort to do in modifying or developing from scratch to achieve the CATNETS simulator.
>
> **Safeness:** inverse of the probability to include bugs in modifying the code.

Scalability. Scalability relates to how well the simulation will work when the size of the simulated ALN increases. Real-world application layer networks are supposed to run several hundreds of agents performing a great number of transactions. Therefore, the efficient simulation of realistic scenario is important.

Framework facilities. Another important aspect to be evaluated is the features that the simulation environment provides to users. We identify a set of relevant features and give a weight of 0.08 to each of them. These features related to are:

> **Probability distribution.** Presence of pseudo-random number generator that ensure good quality (huge period and uncorrelated) random numbers. Presence of several probability distributions (this gives the possibility of generating sequence of events having particular distributions over time).
>
> **Scheduling of future events in simulation.** A very important issue for the simulator is the option of scheduling events at certain points of time. The events are used to examine the behaviour and adaptability of simulated grid systems before an event and after an event. The event scheduling should be able to simulate different dynamic scenarios, which are the main simulation focus in CATNETS.
>
> **Creating GUI and collecting data.** Availability of tools for setting up simulations and for collecting and showing statistics during and after its execution.
>
> **Simulation repetition and parallelism.** Ability of Running several times the same experiment, possibly in parallel machines.

User community. Dimension of the simulator's user community.

4.2.3 Evaluation

The results of evaluation are given in table 4.1. In the following, we motivate the scores given to the two simulators.

features	weight	Repast	OptorSim
Specificity	0.3	1	4
Scalability	0.3	4	3
Random numbers	0.075	4	4
Scheduling	0.075	5	2
GUI and data	0.075	4	4
Repetition and parallelism	0.075	4	3
Community	0.1	4	2
Indicator		3.175	3.275

Table 4.1: OptorSim vs Repast.

4.2.4 Motivation

OptorSim

Specificity. We have already remarked that the ALN model presented in Section has much in common with the model of Data Grid adopted by OptorSim. Moreover, its embedded facilities for parametric flooding of messages in the simulated Data Grid is suitable with few modifications for the implementation of the service/resource negotiation protocols which are essential components of the Catallactic allocation mechanism.

Scalability. **OptorSim** has good scalability performance in term of number of data files requests to satisfy but the number of agents in the Data Grid is restricted to few hundreds.

Probability distribution. OptorSim allows for the specification of the order in which a job requests files. This order is determined by the *Access Pattern* used. Several different access patterns have been chosen for the simulation, allowing the simulation of several types of Grid job. An access pattern can be based on a probability distribution, for example *Uniform*, *Gaussian* or *Zipf*.

Scheduling. OptorSim incorporates strategies for the scheduling of grid Jobs on grid sites. Strategies focus on *where* but not on *when* a job should be scheduled. Interval between job submission is customizable by a simulation parameter. In general, it does not permit the scheduling of specific events at precise time point during the simulation.

GUI and data. OptorSim can be run from the command-line or using a graphical user interface (GUI). A number of statistics are gathered as the simulation runs, including total and individual job times, number of replications, local and remote file accesses, volume of storage filled and percentage of time that sites run jobs. If using the command-line, these are output at the end of the simulation in a hierarchical

way for the whole Grid, individual sites and site components. If the GUI is used, these can also be monitored in real time. A simulation screenshot is presented in Figure 4.2.

Repetition and parallelism. OptorSim is not a distributed simulation framework so does not permit user to run a single simulation in parallel on multiple machines. However, by means of pre-defined scripts it is possible to repeat experiments (possibly varying grid configuration) or run multiple times the same experiment in parallel on clusters of machines.

User Community. OptorSim has been developed and mainly used so far in the framework of the European DataGrid project. However, after the end of Data Grid a number of researchers and students from various institutions worldwide have notified the developers their interest in OptorSim. Some of them have used *OptorSim* as it is or modified for their purposes.
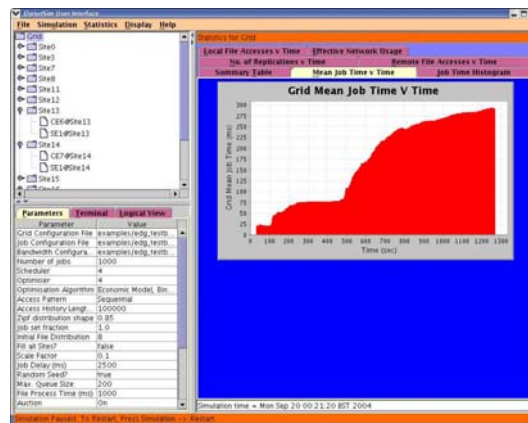

Figure 4.2: OptorSim's simulation screenshot.

Repast

Specificity. Repast is not specific for the simulation of ALN. However, it has special classes to model networks (especially social ones) on which the modelling of ALNs could be based.

Scalability. Most of the application domains where Repast has been used (for example biological application and population dynamics studies) have simulation scenarios where millions of agents are involved.

Random numbers. Repast uses RngPack , a Java package specialised in generating good quality random numbers. It includes 5 pseudo-random number generators

with different period. Repast also includes the Random class that makes it possible to draw random numbers from 13 different probability distributions.

Scheduling. One of the design choices of Repast was driven by the need of analysing the dynamics of systems with huge amount of agents doing things in particular point in time or if specific conditions are satisfied. For this reason Repast has several functionalities to schedule events both at specific and random time points.

GUI and data. Repast has several Graphical facilities. Among them the Wizard class that makes it possible to set up the simulation using a graphical interface and a MovieMaker class that takes images and makes movies out of them. Data collection is handled primarily through the DataRecorder object which can record data from a variety of sources and write that data out in tabular format to a file. Repast provides the user with other way to collect data, for example in the case of parallel simulations. For the input and output of network data specific management classes are provided.

Repetition and parallelism. Repast has a special function, called multirun, to perform several experiments without human management. It function allows the programmer to automatically run a simulation until the specified number of repetitions is reached. Special classes for running simulations in parallel and orchestrate their execution are provided.

User community. Repast has a rather wide community of users and a consolidated development team. The constant interests in the product is witnessed by the number of messages on the Repast interest mailing list (http://sourceforge.net/mailarch ive/ forum .php ?foru m_id =421 5).

4.2.5 Quantitative evaluation outcome

As summarised in Table 4.1 and subsequently motivated OptorSim is the simulator chosen for CATNETS.

Chapter 5

Conclusions

This deliverable has presented the selection process for the simulator to be used in the CATNETS projects. After presenting the requirements for the simulator, the deliverable has described some candidate Grid and general purpose simulators. By means of the following qualitative and quantitative evaluation of these simulators we have come to the conclusion that the Grid simulator OptorSim has the best features to be the starting point for developing the simulator for CATNETS.

The Grid simulator OptorSim was checked against the requirements for the CATNETS simulator presented in Chapter 2. The open issues, a short description of the work to be done, and an evaluation of the complexity of the work (an integer between 1 and 4), are displayed in Table 5.1. The table 5.1 is still preliminary and it will certainly be modified as the project proceeds.

Issue	Description of work	Compl.
ALN model	Data Grid concepts implemented by OptorSim can be mapped into ALN concepts as follows: • **Computing Element** $->$ **Complex Service Provider** • **Storage Element** $->$ **Resource Provider** • **Grid job** $->$ **Complex Service Request** • **Grid file** $->$ **Resource** Basic service providers and basic service requests do not have a corresponding concept in OptorSim. Basic service providers could be obtained by modifying storage elements. Basic service requests could be modelled similarly to resource requests.	2
Service/resource markets	The economic model used by OptorSim for Grid file replication has only one market, where the goods to be traded are data files. The model allows nested negotiations, used by sites that are asked for files they do not store for retrieving them. Nested negotiations could be used to emulate negotiations in the CAT-NETS resource market. These nested negotiations are fired by negotiations in the service market.	2
Bargaining strategy	The bargaining strategy implemented in OptorSim for file selection asks for bids using a flooding mechanism and selects the cheapest bid. Negotiations end after one round. This has to be extended to reflect the CATNETS bargaining model. Bargaining strategy developed in the assessment project could be integrated in OptorSim.	3
Multi-round negotiations	Currently no multi-round negotiations possible, this has to be modified.	4
Event scheduling	No event scheduling is currently implemented in OptorSim. We need two parameters: time and probability. event types: on, off (if no t0 locking we need 2 additionally events), market participation tax event, maximum time for reaching a contract time event, allocation (market clearing) event for centralized market switch_on, switch_off event for dynamicity scenarios	3
Dynamicity	The simulated network is static. Dynamicity should be modelled on time-based events.	3

Table 5.1: Open issues in OptorSim.

Bibliography

[BCC⁺03]   W. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Optorsim - a grid simulator for studying dynamic data replication strategies. *Int. Journal of High Performance Computing Applications*, 17(4), 2003.

[BCCS⁺03]   W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, P. Millar, K. Stockinger, and F. Zini. Evaluation of an Economy-Based Replication Strategy for a Data Grid. In *International Workshop on Agent Based Cluster and Grid Computing at CCGrid2003*, Tokyo, Japan, May 2003. IEEE Computer Society Press.

[BCCS⁺04]   W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, K. Stockinger, and F. Zini. OptorSim v2.0 Installation and User Guide, November 2004. http://edg-wp2.web.cern.ch/edg—wp2/optimization/downloads/v2_0/edg-optorsim/doc/userguide-optorsim.ps.

[CCSM⁺04]   D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini. Analysis of Scheduling and Replica Optimisation Strategies for Data Grids using OptorSim. *Journal of Grid Computing*, 2, 2004.

[CHN03]   N. Collier, T. Howe, and M. North. Onward and Upward: The Transition to Repast 2.0. In *Proc. of the First Annual North American Association for Computational Social and Organizational Science Conference*, Pittsburgh, PA USA, June 2003. Electronic Proceedings.

[edg]   The DataGrid Project. http://www.edg.org/.

[EP00]   T. Eymann and B. Padovan. The Catallaxy as a new Paradigm for the Design of Information Systems. In *Proceedings of the 16th IFIP World Computer Congress, Conference on Intelligent Information Processing*, Beijing, China, August 2000.

[GB02]   N. Gilbert and S. Bankes. Platforms and Methods for Agent-based Modeling. In *Proceedings of the National Academy of Sciences of the USA*,

volume 99, pages 7197–7198, Washington, DC, USA, 2002. National Academy of Sciences of the USA.

[HBKC89] F. A. Hayek, W.W. Bartley, P.G. Klein, and B. Caldwell. *The collected works of F.A. Hayek*. University of Chicago Press, Chicago, 1989.

[Jai91] Raj Jain. *The Art of Computer Systems Preformance Analysis*. John Wiley and Sons, Inc, 1991.

[jas] JAS web site. `http://jaslibrary.sourceforge.net`.

[lcg] LHC Computing Grid. `http://lcg.web.cern.ch/LCG/`.

[opt] Optorsim web site. `http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html`.

[rep] Repast web site. `http://repast.sourceforge.net`.

[SD02] A. Serenko and B. Detlor. Agent Toolkits: A General Overview of the Market and an Assessment of Instructor Satisfaction with Utilizing Toolkits in the Classroom. Technical Report Working Paper 455, McMaster University, Hamilton, Ontario, Canada, 2002.

[SPBT05] Anthony Sulistio, Gokul Poduvaly, Rajkumar Buyya, and Chen-Khong Tham. Constructing A Grid Simulation with Differentiated Network Service Using GridSim. In *Proc. of the 6th International Conference on Internet Computing (ICOMP'05)*, Las Vegas, USA, June 2005.

[swa] SWARM Web site. `http://www.swarm.org`.

[SYB04] Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software Practice and Experience*, 34(7), 2004.

[TH03] R. Tobias and C. Hofmann. Evaluation of Free Java-libraries for Social-scientific Agent Based Simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), 2003.

[WP105] WP1. Environmental Analysis of Application Layer Networks. Technical Report WP1 - D1, CATNETS EU IST-FP6-003769, 2005.

[WP405] WP4. Metrics Specification. Technical Report WP4 - D1, CATNETS EU IST-FP6-003769, 2005.

In this paper the requirements for an ALN
simulation environment are analysed, as needed
in the CATNETS Project. A number of grid and
general purpose simulators are evaluated
regarding the identified requirements for
simulating economical resource allocation
mechanisms in ALNs. Subsequently a suitable
simulator is chosen for usage in the CATNETS
project.