Lehrstuhl für
Wirtschaftsinformatik
Information Systems
Management

No. 4
2005

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Gianfranco Giulioni, Floriano Zini

Analysis and selection of the Simulation Environment

Bayreuth Reports on Information Systems Management

UNIVERSITÄT
BAYREUTH

ISSN 1864-9300

Authors:

Gianfranco Giulioni, Floriano Zini

Information Systems and Management Working Paper Series

Edited by:

Prof. Dr. Torsten Eymann

Managing Assistant and Contact:

Raimund Matros
Universität Bayreuth
Lehrstuhl für Wirtschaftsinformatik (BWL VII)
Prof. Dr. Torsten Eymann
Universitätsstrasse 30
95447 Bayreuth
Germany

Email: raimund.matros@uni-bayreuth.de

ISSN 1864-9300

Analisys and Selection of the Simulation Environment

WP4: Simulator
Gianfranco Giulioni - Università delle Marche
Floriano Zini - ITC-irst

Executive Summary.
CATNETS EU IST-FP6-003769 Project Deliverable D4.1
This document provides the initial report of the Simulation work package (Work Package 4, WP4) of the CATNETS project. It contains an analisys of the requirements for a simulation tool to be used in CATNETS and an evaluation of a number of grid and general purpose simulators with respect to the selected requirements. A reasoned choice of a suitable simulator is performed based on the evaluation conducted.

Document Id.	CATNETS/2005/D4.1/v1.0
Project	CATNETS EU IST-FP6-003769
Date	Mar 18th, 2005
Distribution	internal

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
Contact person: Torsten Eymann
E-mail: catnets@uni-bayreuth.de

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

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

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

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
Contact person: Omer F. Rana
E-mail: o.f.rana@cs.cardiff.ac.uk

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

Changes

Version	Date	Author	Changes
0.1	07.02.05	Gianfranco Giulioni	First draft
0.2	15.02.05	Floriano Zini	Formatting using CATNETS style
0.3	28.02.05	Gianfranco Giulioni	Second draft
0.4	09.03.05	Floriano Zini	Improvements of ALN reqs
1.0	18.03.05	Gianfranco Giulioni - Floriano Zini	Final version to be submitted to reviewers

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 provisioning of services and resources to network clients. This should be done by a scalable, dynamic, and adaptable allocation mechanism. 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 for both P2P and Grid networks.

The first goal of CATNETS is to evaluate the proposed allocation mechanism, based on the economic paradigm of Catallaxy[HBKC89, EP00]. Evaluation is planned to be done in a simulated ALN. Simulation 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. Comparison of the Catallactic mechanism with other approaches in a simulated environment will allow detailed investigation of aspect such as scalability, topology influence, or connection reliability.

The second goal of CATNETS is to produce a “proof-of-concept” prototype upon a real ALN. The use of simulation will give us theoretical and practical hints about how to build a prototype application based on the concept of ALN.

The selection of a suitable simulation tool is fundamental. In order to do so, this document first introduces the salient features of the scenarios to be simulated and the requirements for a simulator for such scenarios. The rest of the document describes some grid and general purpose simulators and assess them with respect to the selected requirements. The motivated choice of the simulation tool concludes the report.

Chapter 2

Simulator Requirements

Simulation is a common and useful technique for the analysis, design and evaluation of computer systems[]. 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 in a controlled way.

Since our first goal is to make a thorough comparison between Catallaxy and traditional service and resource allocation mechanism and our second goal is to build a prototypical application for which we need to elicitate design requirements, simulation is appropriate for our purposes.

The main criteria that drives the selection of the requirements for the CATNETS simulator is independence from applications. In fact, the CATNETS simulator should be able to execute models which do not depend on particular types of P2P or grid systems but are abstract enough to derive general conclusions about the evaluated service and resource allocation mechanisms. This criteria leaded us to identify four types of functional requirements for the simulator:

- requirements related to the concept of ALN;
- requirements related to the concept of the service and resource allocation mechanism;
- requirements related to the concepts of scalability;
- requirements related to the concept of evaluation metric.

We also identified two non-functional requirements to be taken into consideration:

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

- the past experience of the consortium members in the realisation of simulators is a key factor that can speed up the realisation of the CATNETS simulator.

In the following section we go into the four types of functional requirements.

2.1 Application Layer Networks

The term "Application Layer Networks" (ALN) 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 network from the user's view. This concept can be useful in order to study in general this network systems but it needs to be precisely defined.

We adopt a model where an ALN consists of several *sites*, each of which may provide services and resources for submitted *jobs*. In each site are located zero or more *Service Providers* and zero or more *Resources*. Each resource has an associated *Resource Provider* Service providers execute part of jobs by exploiting resources. ALN sites are connected by *Network Links*, each of which has a certain bandwidth.

In our model a job is specified by the *set* of services it needs to use. Jobs are submitted to the ALN by *clients*. A service is specified by the *set* of resources which are requested in order that the service can be provided.

There are a number of parameters that characterise a ALN specified by the model above. They are:

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

Service distribution. Services 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 of resources and services can vary over time.

Resource diversity. There can be several types of resources, from commodity resources to highly specialized, unique resources.

Network cost. Accessing remote services or resources can have a cost.

Usage Patterns. Clients may request the same services recurrently or request different services each time.

2.2 Service and resource allocation mechanism

The key idea proposed by the CATNETS project is to adopt a 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 []. 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 clients and service providers trade for services. The second is called *resource market* and is the place where service providers and resource providers trade for resources. This reference scenario is illustrated in Figure 2.1.

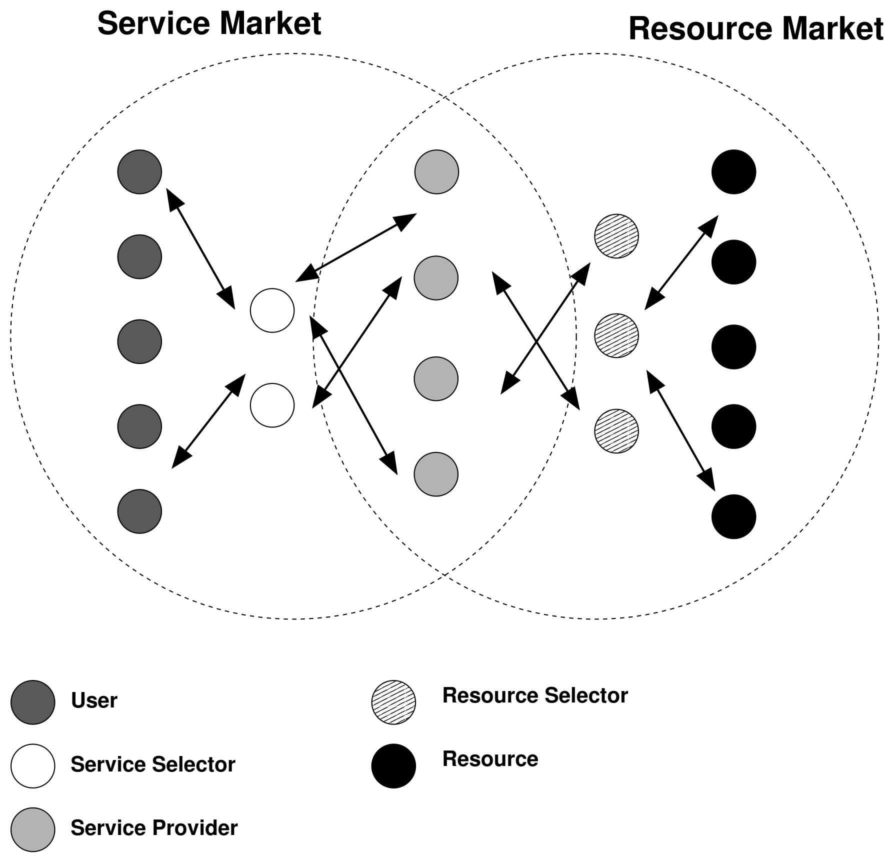


Figure 2.1: Service and resource markets.

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

Client. Clients need their jobs to be executed by purchasing services available on the service market;

Service Selector. It acts on behalf of one ore more clients and provides them with ”best” services that are needed to execute their jobs. In order to do so, it negotiates with service providers;

Service Provider. Its goal is to sell a service to clients 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*:

Service Provider. Each service needs a set of technical resources. A service provider needs to optimally purchase these resources on the resource market;

Resource Selector. It acts on behalf of one ore more service providers and its goal is to allocate ”best” resources that are necessary for the provision of the service. In order to do so, it negotiates with resource providers;

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

Service selection can be centralised or with various levels of decentralisation. In the most centralised scenario, there is only one service selector that acts on behalf of all the clients. In the most decentralised scenarion, each client plays also the role of service selector. Similarly, in the resource market the scenarions can range from just one resource selector (centralised approach) to the case where the role can be played by the service providers (total decentralisation).

2.2.2 Negotiation protocol

A fundamental components of each of the two markets is the protocol used for negotiation among market’s players. As stated in the CATNETS Deliverable D1.1[], in the context of the service and resource markets the use of auctions is an efficient way to allocate these services and resource, as well as to determine their prices.

The adoption of an auction model implies that the negotiation protocol is based on message passing. Players in the market need to exchange messages related to call for bids, bids, acceptance or refusal of prices. This need to be easily modelled in the simulation environment.

2.3 Scalability

The definition of scalability is not unique and can cover different aspects of the system. In general all the definitions are related to the performance of the system and some of them take care of different aspects such as code maintanabilility, fault tolerance and availability of programming staff. The opinion that scalability is basically a measure of performance can be criticized because it could be possible to find something that perform poorly but scale well.

We agree with the opinion that scalability is tied to the amount of resouces needed to make working a system of a given size (a certain number of clients, services and resouces). In particular what is interesting is how the required resources growth with the size of a system. In a non scalable system, the rate of growth of needed resources is increasing while in a scalable one it is constant or decreasing. Put in onother way in a non scalable environment an increase of the system size causes an increase of the average resouces needed (this is a relative measure: the ratio between the amount of resources and the a proxy of the system size.)

For overlay networks, scalability is often refered to as the number of users in a given system or community. The larger it is the larger is the amount of resouces available for exploitation. This interpretation of scalability assume that the ability to find the resouce remains high when the system growth.

According to this view a simulator that aim to evaluate the performance of a given ovelay network should include a very high number of nodes. As pointed out before the CATNETS project aim to investigate how economic mechanisms can be used to improve the performance of ALNs. In this case, an alternative view to the number of nodes could be to have an elevate rate of service and resource requests that are satisfied. This give us some hint about the features of the CATNETS simulator: it should contemplate the possibility to incorporate a very large number of agents or alternatively a huge amount of service and resourcse requests.

It is natural to think that there is a strong trade off between the scalability of the simulator and its accuracy or level of details it can manage. From this observation we can confirm the previous requirement for the CATNETS simulator: the ALN and alocation mechanism model need to be abstract and avoid including too details.

2.4 Metrics

In order to evaluate performances of different scenarios about different ALN parameters (resouce distribution, service distribution, configuration dynamism, resouce diversity, network costs and usage patterns) and different market structures on the services and resouces markets the simulator should be able to collect data and calculate metrics.

Some of these metrics are proposed in the deliverable D1.1[] and, for the sake of completeness, described again below.

From microeconomic theory we borrow the concept of **allocative efficiency**. A situation is efficient in the allocative sense if moving from this situation the utility of at least one agent decreases.

Evaluating "how" the actual system outcome is far away from this situation is not an easy task. There are two possibilities if we know the reservation values of agents. Let v_i and s_j be the reservations value of buyers and sellers and with p the price payed or charged.

1. Then a possible measure is the overall welfare:

$$S = \sum_{i=1}^{I}(v_i - p_i) + \sum_{j=1}^{J}(p_j - s_j)$$

2. From the buyers and sellers reservation prices we can compute aggregate demandad and supply curves. The procedure is as follows. For each price level p we can compute following sums:

$$D(p) = \sum_{\{I|v_i \geq p\}} q_i$$

$$S(p) = \sum_{\{J|s_j \leq p\}} q_j$$

   Where $\{I|v_i \geq p\}$ is the set of clients that satisfy the condition $v_i \geq p$ and $\{J|v_j \geq p\}$ is the set of servers that satisfy the condition $s_j \leq p$

   These quantities are respectively the aggregate demand and aggregate supply. If reservation prices are heterogeneous among agents they are respectively decreasing and increasing by construction so they intersect in a point. This intersection point give us two very important theoretical values: the theoretical equilibrium price p^* and the theoretical quantity X^*. Here we point our attention to quantity while prices will be discussed later.

   Standar microeconomi theory demostrate that under non restrictive assumpions X^* is the Pareto efficient quantity. Now from the simulation we can get the sum of the exchanged quantity Q. This quantity is surely less than the maximim possible quantity X^*.[1]

   From the economic point of view the most important metris to evaluate allocative efficiency of a system is:

$$eff = \frac{Q}{X^*}$$

[1]Microeconomic theory identify the condition under which $Q < X^*$. They are alla situation where a centralized market doesn't exist, or a market for some good doesn't exist at all: public goods, externalities, asymetric information and so on.

Knowing the theoretical values of prices we can compute a metrics to evaluate **informational efficiency**. nities. The mean square deviation of actual price from rational expectation price p^* is proposed as a metrics for this goal.

Other possible criteria to evaluate a system performance are:

- **Revenue maximization:**. To evaluate this concept we can use the following metric:
$$\sqrt{\frac{\sum_i (a_i - \pi_i)^2}{n}}$$
where $\pi_i = |r_i - p_i|$ are theoretical payoff (r_i is the reservation price of clients, servers or service providers), a_i the realized payoff and n is the number of agents.
- **Correctness:** evaluated as the average number of times that a mechanism has approached the optimal allocation p^* and X^*.
- **Stability:** in a decentralized bilateral exchange setting, a measure of stability is the coefficient of convergence of prices to equilibrium values, that is defined as the ratio between standard deviation of actual prices and the predicted equilibrium price. The latter in turn is given by the intersection of demand and supply curves. Related to this metrics is the "time" of convergence i.e. time elapsed until the aforementioned ratio has shrunk to zero.
- **Incentive Feasibility:** in many mechanisms market agent interactions lead to strategic behaviour. If an agent knows that he can influence final price - and then his own payoff - he can report bids untruthfully. Thus, a mechanisms have to be evaluated with respect to the propensity of agents to report the truth values of goods in transactions. Actually, convergence to equilibrium can be attained also if traders use strategy of untruthful report their preferences, determining inefficiency (as a result some traders can stay out market also if they are willing to trade). A measures for the foregoing feature, by evaluating the ratio of expected gains from trade across all agents in equilibrium and the expected gains from trade across all trades in equilibrium assuming that all agents behave as price takers. Incentive compatible mechanisms must be such that this ratio to equals 1.
- **Communication Costs:** Hayek works out the idea that in standard environments, the Walrasian mechanism is "informationally efficient", in that it realizes Pareto efficient allocations with the least amount of communication. The Walrasian mechanism involves only the announcement of prices, along with the allocation, which is much more economical than full revelation of agents' preferences. [NS01] examine communication problem establishing a lower bound of numbers of message needed to ensure efficient outcome. Thus a metrics for this feature is the number of bids in transaction needed to approach the efficient outcome.

- **Computation Costs** [Axt99] studies the complexity of exchange and establishes a lower bound of interactions between agents needed to ensure an efficient outcome. In this respect a metrics is the number of interactions which allows allocative efficiency. In general, this measure is exponential in the number of commodities and agents.

Chapter 3

Simulators

The massive increase of network application usage generated a lot of interest and effort in understanding the way to make then 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 on each of them.

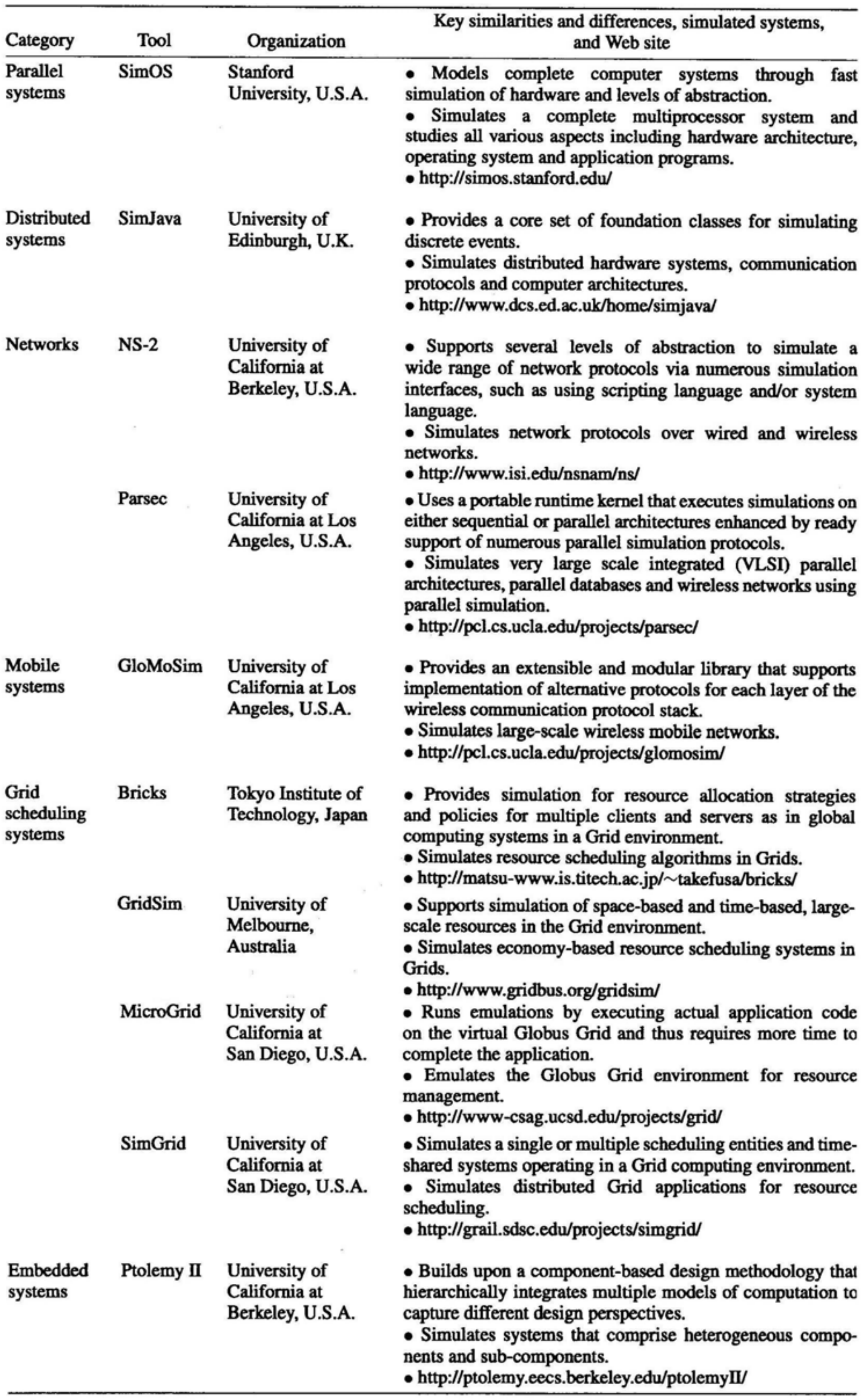
It is quite clear that a comprehensive evaluation of the majority of existing simulators is very difficulto to achieve. Moreover, by rely on assessments done by others we have the problem that the requirements chosen by eveluation do not match with ours. For these reasons we decided to assess simulators over the requirements presented in Chapter [] 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 somulstors on which we have less deep knowledge.

3.1 The consortium experience

3.1.1 Catnet simulator

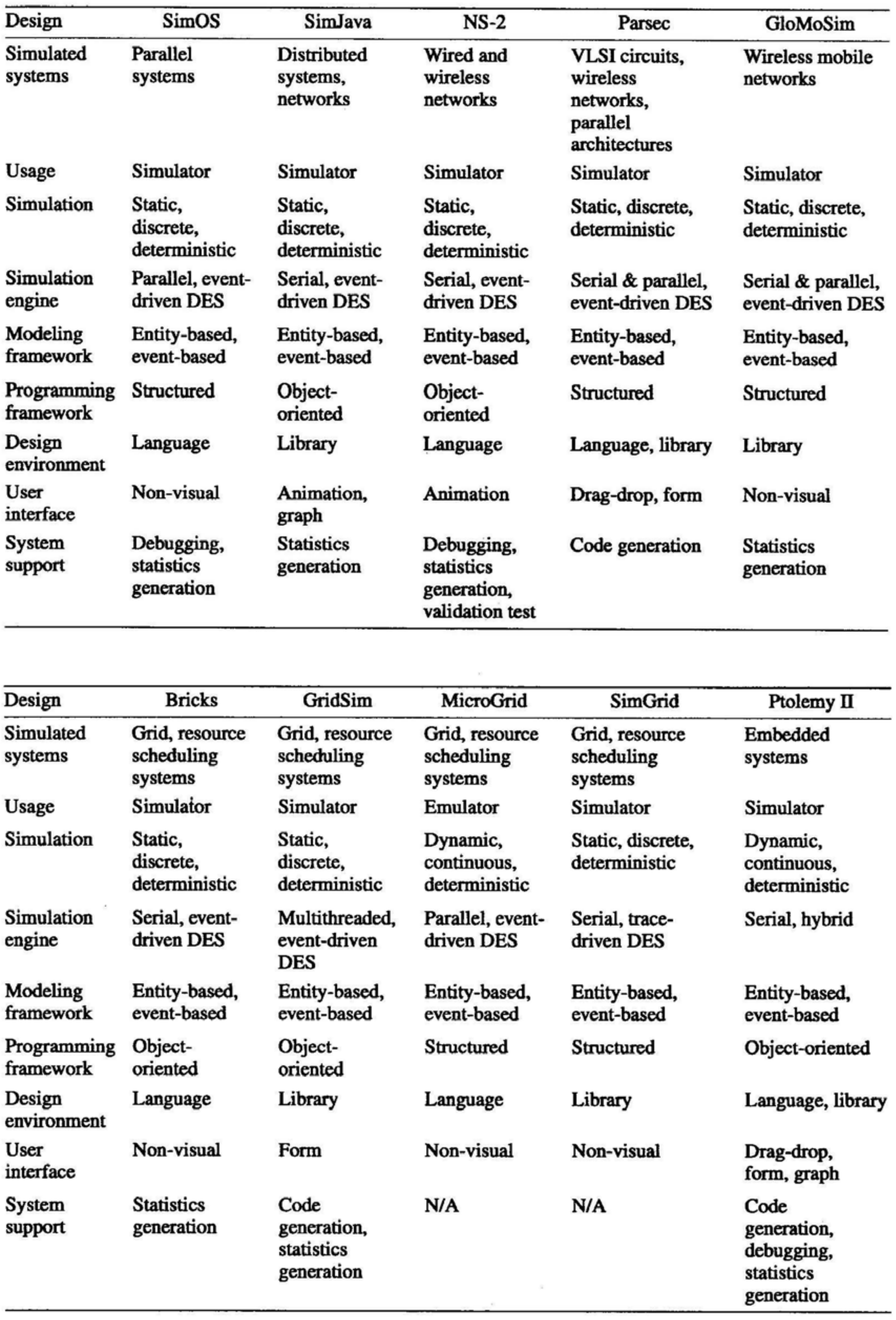
In the CATNET assesment project a simulator was developed to evaluate the behavior of a P2P system with the catallactic coordination mechanism. CATNET is a simulator for an application layer network, which allows creating different types of agents to form a network. This simulator is implemented on top of the J-Sim network simulator.[1]

[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.



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].

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 instance. 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 build 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 the resources
- The initial prices of Clients, Service Copies, and Resource agents
- initial budget of Clients
- the type of control mechanism (baseline or Catallactic)

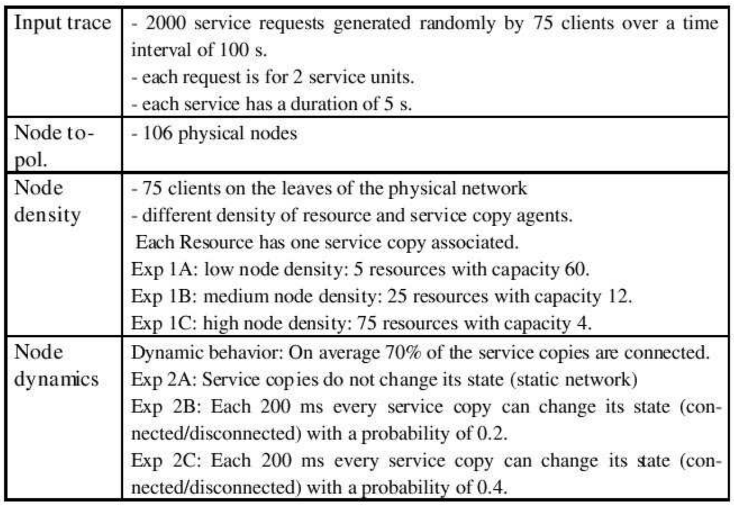
The CATNET simulator allows to vary two important parameters:[2]

[2]Though different in many particular mechanisms, real world applications (multimedia content distribution networks (for instance Akamai), Grid implementations, and Peer-to-Peer systems (for instance Gnutella)) can be mapped to the two dimensional design space given by these two features.

1. the node dynamics;
2. the node density of the application layer network.

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 correspoding to three different levels of node dynamics (null, medium and high) and three levels of node density (low, medium and high). [3] 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 to-pol.	- 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 dine with the CATNET simulator.

3.1.2 OptorSim

OptorSim ([CCSM$^+$ng, BCC$^+$03a, wsa]) is a joint effort of ITC-irst, University of Glasgow and CER. It is a Grid simula@miscedg, title = The DataGrid Project, note = http://www.edg.org/ tor that has been developed in the framework of the European DataGrid (EDG) [edg] in order to explore by simulation the behaviour of different data replication algorithms in several grid scenarios. It has been shown [RF01, BCC$^+$03b]

[3] 18 basic experiments was conducted in total.

that data replication - the process of placing copies of files at different sites - is an important mechanism for reducing data access times and hence improving overall resource usage.

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

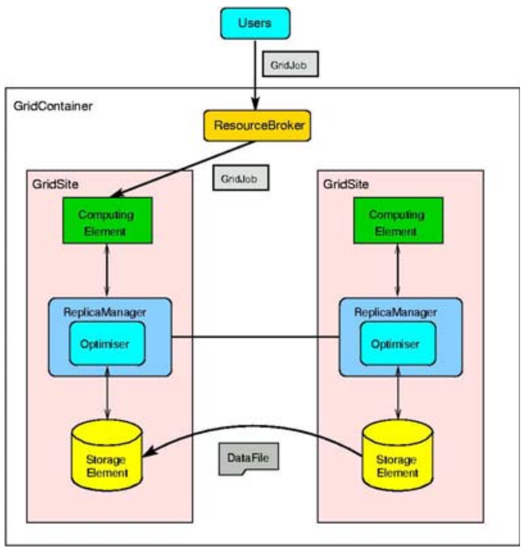


Figure 3.3: OptorSim Architecture.

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 *AdvancedGridTime* speeds up the running of the simulation considerably, whereas *SimpleGridTime* may be desirable for demonstration or other purposes.

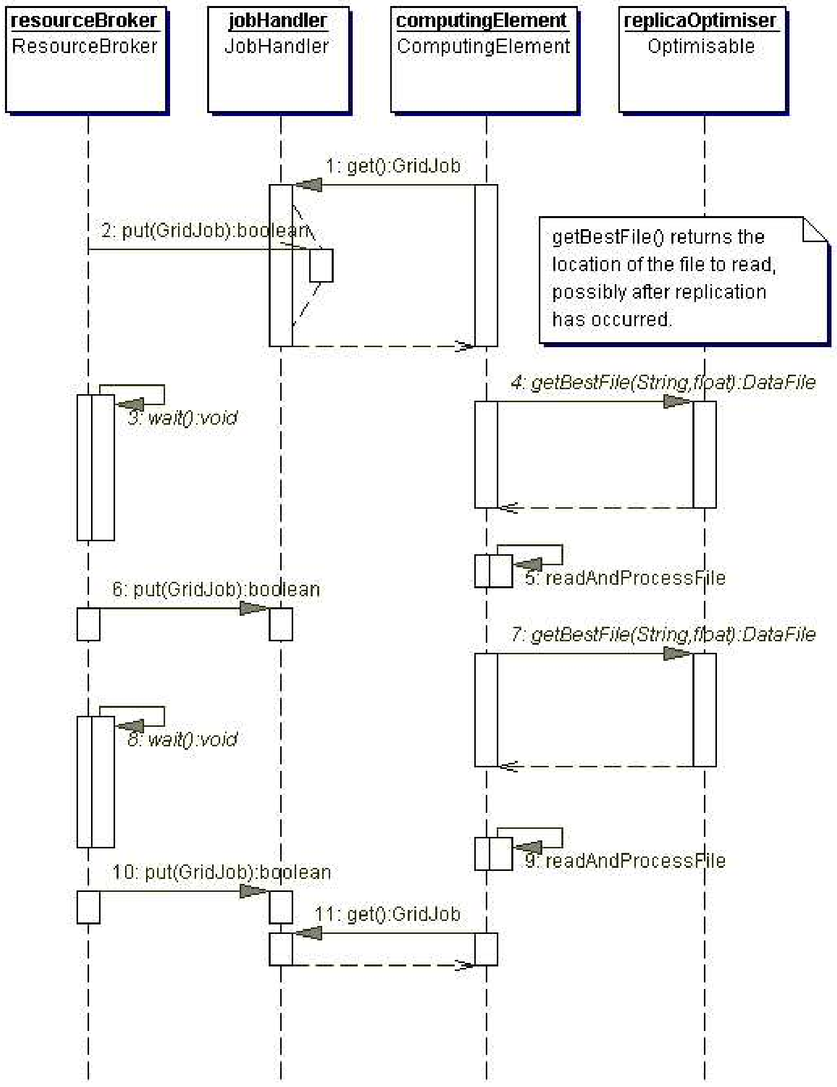


Figure 3.4: Sequence diagram of the Resource Broker and Computing Element threads.

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[4] Data Challenge 2002 testbed (Figure 3.5) and the LCG 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 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:

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

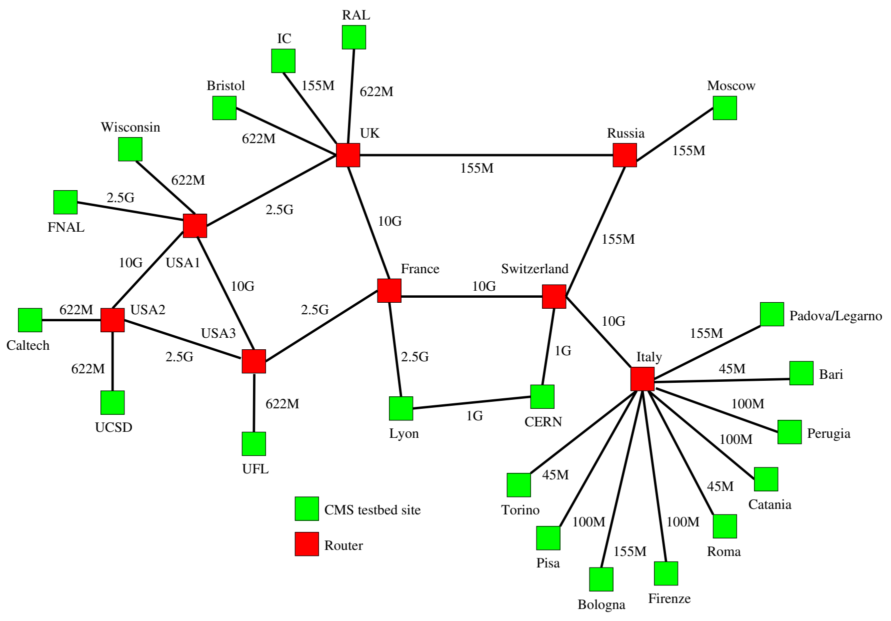


Figure 3.5: CMS Data Challenge 2002 grid topology.

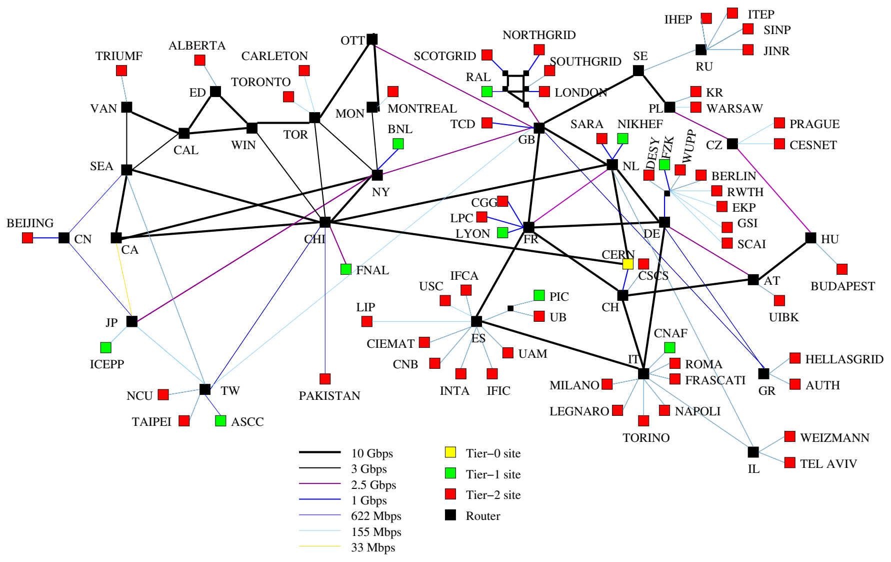


Figure 3.6: LCG August 2004 grid topology.

- $D/\langle SE \rangle$. 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.

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

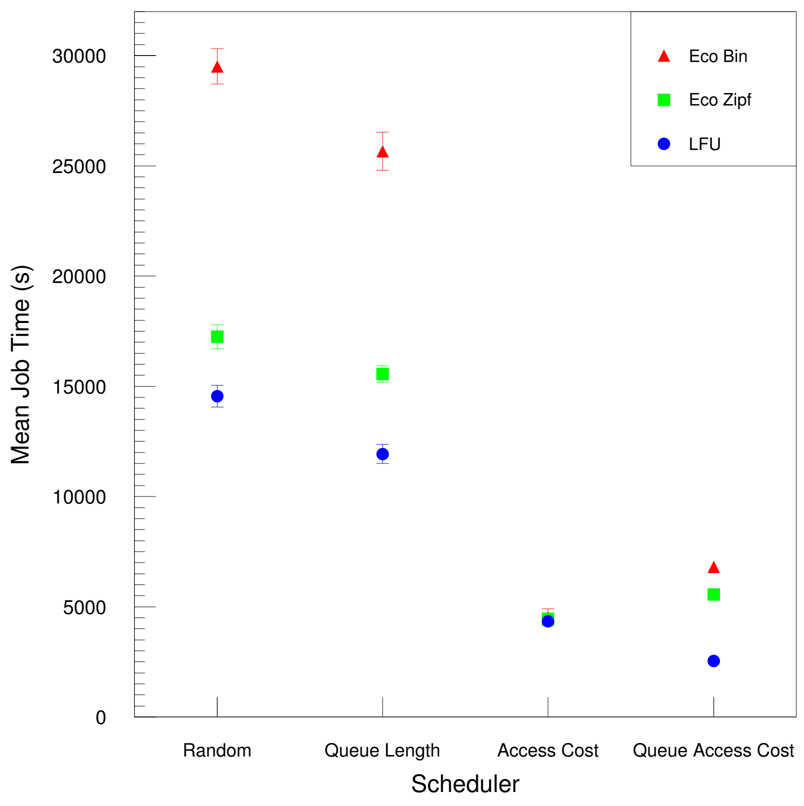


Figure 3.7: Mean job time for scheduling and replication algorithms in CMS 2002 testbed.

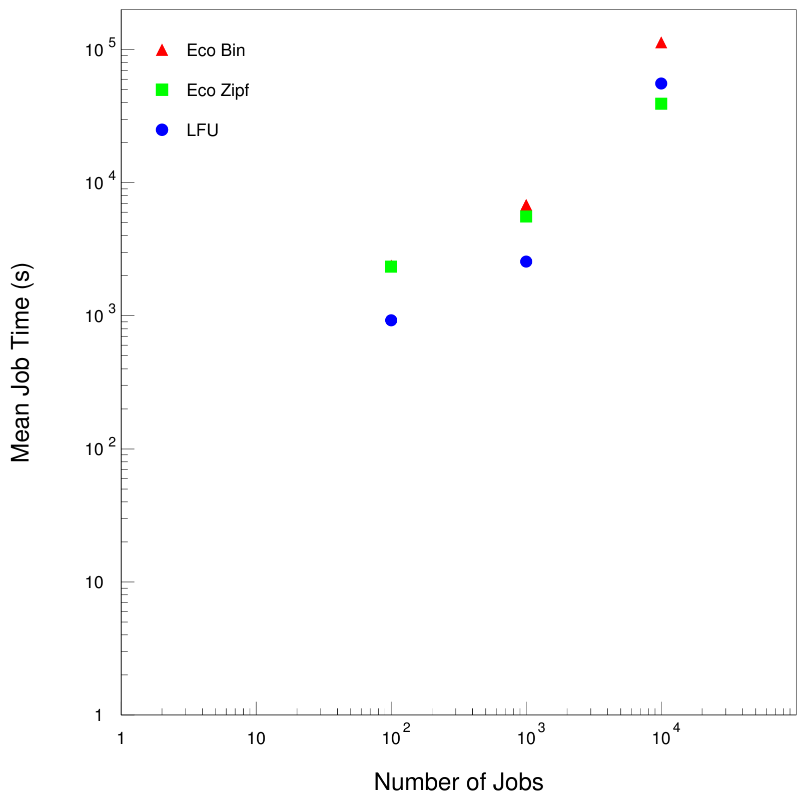


Figure 3.8: Mean job time for increasing number of jobs in CMS 2002 testbed.

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

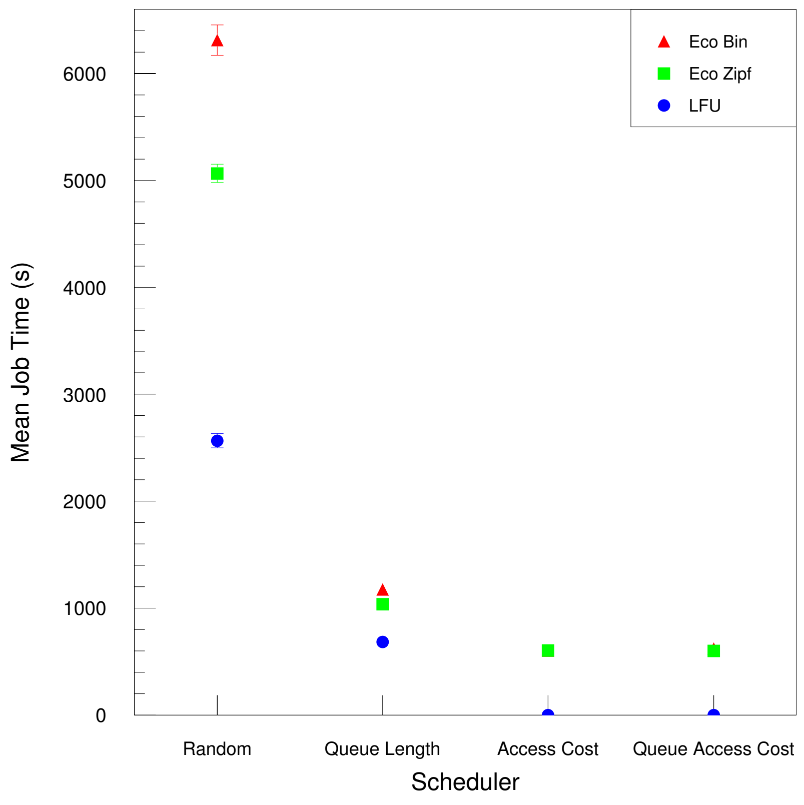


Figure 3.9: Mean job time for scheduling and replication algorithms in LCG August 2004 testbed.

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 [wsb] is the first simulator adopting an interesting simulation design. According to this design a simulation is organised in two levels: the first level concerns the interaction between the user and the simulator (called the observer) while the second concerns the modelling of the scenario to be simulated.

The possibility to interact with the model is particularly useful in the preliminary stage of simulation. In this phase one have to check the behavior of the model to discover eventual bugs. Of course the presence of 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).

Both the observer and the model are organized in two phases. In the first one the agents in the model are designed defining the actions they can perform and how they can do it and finally they are created. In the second phase the dynamics of the model is designed giving to each agent its own schedule.

This procedure was implemented for the first time in 1995 in the simulator developed by the Santa Fe Institute and known as SWARM. The simulator was originally a library written in objective C, but in recent years it was rewritten to be used with the java language. So actually Swarm software comprises a set of code libraries which enable simulations of agent based models to be written in the Objective-C or Java computer languages. These libraries will work on a very wide range of computer platforms.

After a long period of testing a new version of SWARM (2.2) was released in February 2005. A notable characteristic is the ability to collect data in the binary format called HDF5.

Swarm has 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 anaysis

Economics: Stock Market simulation

Political Science: Political Party formation simulation

Geography: Traffic pattern analysis, Land use analysis

Military: Weapon Deployment analysis

Some interesting applications of SWARM in computer science and industry are:

- The CAR Group at the University of Michigan Program for the Study of Complex Systems has developed a tool called "Drone" that can be used with Swarm or with other simulation packages to do multiple runs of a simulation while varying the inputs automatically. The CAR Group consists of Michael Cohen, Robert Axelrod, and Rick Riolo.
- General Electric's Imperishable Networking group used Swarm to model the evolution of complexity on an active network.
- Fabrice Chantemargue implemented a model of Implicit cooperation and Antagonism in Multi-Agent Systems in Swarm. He has an enhanced application that uses the Vision library to help model this antagonism.
- Jim Clark at McGill University is porting a simulation of an MIMD parallel computer via "processor agents" moving around in a data space to do load balancing on those processors. He has provided a postscript paper entitled "A Model for Natural and Artificial MIMD Systems" describing the model.

RePast

The Recursive Porous Agent Simulation Toolkit (Repast)[wsc] is a free 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 agent-based modeling toolkit. Repast is differentiated 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 [SD, GB02, TH03].

At its heart, Repast toolkit version 3.0 can be thought of as a specification for agent-based modeling services or functions. There are three concrete implementations of this conceptual specification. Naturally, 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.

The last release (Repast 3.0 released November 15, 2004) has a variety of features including the following:

- the toolkit gives users complete flexibility as to how they specify the properties and behaviors of agents.

- Repast is fully object-oriented.
- Repast includes a fully concurrent discrete event scheduler. This scheduler supports both sequential and parallel discrete event operations.
- Repast offers built-in simulation results logging and graphing tools.
- Repast has automated Monte Carlo simulation framework.
- Repast provides a range of two-dimensional agent environments and visualizations.
- Repast allows users to dynamically access and modify agent properties, agent behavioral equations, and model properties at run time.
- Repast includes libraries for genetic algorithms, neural networks, random number generation, and specialized mathematics.
- Repast includes built-in systems dynamics modeling.
- Repast has 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.

RePast has some application in pc networks [NH].

JAS

JAS (Java Agent-based Simulation library)[wsd] 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.
- Different time unit management (ticks, seconds, minutes, days...).
- The real time engine is able to fire events using the real computer timer. Useful for emulation models.
- 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.
- The MultiRun class manages looped model run for automatic parameters calibration.
- The statistical package, based on the cern.jet package, with file and database I/O features.

The Latest release (JAS 1.0) has been released in May 2004.

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, might satisfi the requirements for the CATNETS simulator.

3.2.1 P2Psim

p2psim is a freeware, multi-threaded, discrete event simulator to evaluate, investigate, and explore peer-to-peer (p2p) protocols. p2psim runs in Linux and FreeBSD. p2psim is part of the IRIS project. The goals of this simulators:

1. to make understanding peer-to-peer protocol source code easy;
2. to make comparing different protocols convenient
3. to have reasonable performance.

Because p2psim uses threads, implementations look like algorithm pseudo-code, which makes them easy to comprehend.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.

p2psim already supports Chord, Koorde, Kelips, Tapestry, and Kademlia. These implementations are specific to p2psim. They consist of substantially fewer lines of code than the real implementations.

3.2.2 Planetsim

PlanetSim is an object oriented simulation framework for overlay networks and services. This framework presents 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 like Chord or Pastry, or creating and testing new services (DHT, CAST, DOLR, etc) on top of existing overlays.

PlanetSim also aims to enable a smooth transition from simulation code to experimentation code running in the Internet. Because of this, 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 to reduce complexity and smooth the learning curve in our framework. The code is optimised to enable scalable simulations in reasonable time.

Last release: version 1.0 date July 09, 2004.

3.2.3 Peersim

Peer-to-peer systems can reach huge dimensions such as millions of nodes, which typically join and leave continuously. These properties are very challenging to deal with. Evaluating a new protocol in a real environment, especially in its early stages of development, is not feasible. There are distributed planetary-scale open platforms (e.g., PlanetLab) to develop and deploy network services, but these solutions do not include more than about 440 nodes. Thus, for large-scale systems, a scalable simulation testbed is mandatory.

Peersim has been developed with extreme 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. It is the evolution of the anthill project started in 2001. Peersim is written in Java.

3.2.4 Diet Platform

The DIET Agents 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 project finished 1 July 2003, after which the DIET Agents platform has been released as Open Source. 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.

Chapter 4

Discussion

In this chapter we assess the simulators presented in the previous chapter with respect to the requirements described in Chapter 2.

4.1 Requirements related to the concept of ALN

Two systems can naturally simultate an ALN corresponding to the model presented in Chapter 2. They are OptorSim, developed 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. In both simulators the building blocks for the specification of the ALNs to be simulated corresponds quite well to the principal components of the ALN model presented in Chapter 2. All the other simulators described in Chapter 3 are not tailored for the specification an ALN having the described model because the building blocks for the definition of a simuilation configuration are too low level.

Considering the ALN parameters presented in Chapter 2, Optorsim can model ALNs where there is resource and service distribution, network cost and usage patters, but are not dynamic. Instead, the Catnet simulator gives the uses the possibility of simulating configuration dynamism.

4.2 Requirements related to the concept of the service and resource allocation mechanism

Optorsim seems to be the simulator that best fullfill these requirements. In fact, it allows for the simulation of multiple auctions that can are used for negotiation of data files in the simulted grid.

The other simulators do not provide auctions as first level objects. This means that, adopting one of these simulators, auction protocol had to be implemented from scratch.

4.3 Requirements related to the concept of scalability

Agent based simulators presented in Section 3.1.3 are higly scalable [Set03] because of their architecture. The simulation engine is basically time stepped as in the peersim case. At each time step one, more all the agents can be selected in a sequential or random way to perform the particular tasks scheduled by the programmer. The main fact that allow to this simulators to be highly scalable is the use of message passing instead ot using threads. the following sentence reported fom the swarm mailng list is significative to evaluate scalability: *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.*

To achive scalability the **peersim** engine adopts a time-stepped simulation model instead of more complex and expensive event-based architecture. At each time step, all nodes in the system are selected in a random order, and a callback method is invoked on each of the protocols included in that node. In this way, all protocol instances 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. The fail-fast, resource constrained execution of kernel functions lets systems gracefully cope with overload and failure. Feedback provided by the kernel enables agents to adapt to changing conditions and overload.

The current implementation of **OptorSim** allows for the simulation of grids having up to 70 sites. In each site there can be up to 4 agents modelling grid site's components. The number auctions for data file that can be performed contemporary is around few hundreds.

4.4 Requirements related to the concept of evaluation metric

The metrics presented in Section 2.4 are directly derived from economic theory and mechanisms for their calculations are not included in any of the simulators presented above.

Chapter 5

Conclusions

The main task of WP4 in the first six months of the project was to evaluate and select a simulation environment. A relevant effort was to collect information on a number of candidate simulation tools to have the wide view of the field.

The result of this investigation lead to a bunch of alternatives that could in general be collocated in a two dimensional space: scalability and specificity. A simulator is labeled as specific if it was build to analyze particular situations and general purpose if it can evaluate the performances of a generic network.

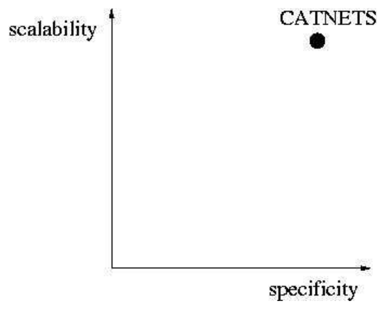


Figure 5.1: Scalability vs specificity

Second point was to investigate and to understand in coordination with WP1 and WP3 the requirement of the CATNETS simulator with respect to the two dimensions we identified above. The choice of the level of specificity is implicit in the CATNETS project: it has to analyze the efficiency of a system with respect to different transactions mechanisms (centralized vs decentralized and various decentralized). 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 transaction mechanisms.

The decision on the scalability was facilitated by the opinion of the WP3 components. From these exchanges of opinion we formed the idea that the CATNETS simulator should be highly scalable and have a lightweight structure.

The subsequent effort was to collect data from the existing highly scalable simulators. We realized that the simulators particularly interesting for the CATNETS project was peersim, the DIET platform and the agent based simulators (Swarm and RePast). These simulators have the possibility to manage a large amount of agents.

Our investigation of these simulators pointed out the disadvantage of peersim that have a completely different specialization from that required by the CATNETS project and of DIET that is more an emulator rather than a simulator.

On the side of specificity, the simulator Optorsim provides building blocks for a simulation scenatio that correspond very well to the principal components of an ALN and a servece or resource allocation mechanisms. Since it simulates grid with less the 100 sites, its drawback could be scalability. However, it allows for the simultaneus management of several hundreds of user's job.

In conclusion, we see two possibilities. The first is to use the an agent based simulator (swarm and repast) that are used in disciplines like biology, social systems and so on to simulate systems with a huge amount of agents. The drawback of this kind of simulators is that they are general purpose and must be specialized for CATNETS goals. The second one is optorSim that already include some concept we need (for instance an auction mechanism) so that it has a low cost to adapt but it could have some problem for scalability. A choice among these two simulator need some more investigation.

Bibliography

[Axt99] Rob Axtell. The complexity of exchange. Technical report, Society for Computational Economics, March 1999.

[BCC$^+$03a] 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, 2003.

[BCC$^+$03b] W. H. 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 v1.0 Installation and User Guide, February 2004. http://edg-wp2.web.cern.ch/edg-wp2/optimization/downloads/v1_0/edg-optorsim/doc/userguide-optorsim.ps.

[CCSM$^+$ng] 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*, forthcoming.

[CHN03] N. Collier, T. Howe, and M. North. Onward and upward: The transition to repast 2.0. In *Proceedings of the First Annual North American Association for Computational Social and Organizational Science Conference, Electronic Proceedings, Pittsburgh, PA USA*. National Academy of Sciences of the USA, Washington, DC, USA, 2003. vol. 99, suppl. 3.

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

[EP00] T. Eymann and B. Padovan. The catallaxy as a new paradigm for the design of informa- tion systems. Technical report, Proceedings of The World Computer Congress 2000 of the International Federation for Information Processing, 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*, pages 7197–7198. National Academy of Sciences of the USA, Washington, DC, USA, 2002. vol. 99, suppl. 3.

[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.

[lcg] GDB Resource Allocation and Planning. http://lcg-computing-fabric.web.cern.ch/LCG-Computing-Fabric/GDB_reosur%ce_allocation_planning.htm .

[NH] Michael J. North and Cynthia S. Hood. Multi-agent modeling of high performance computing cluster users. Proceedings of the 2004 Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation, New York, NY USA, 10 pgs. (July 2004). http://www.agents.cs.nott.ac.uk/events/mamabs04/.Papers/05-North.pdf.

[NS01] Noam Nisan and Ilya Segal. The communication complexity of efficient allocation problems. Unpublished Paper, November 2001.

[RF01] K. Ranganathan and I. Foster. Identifying Dynamic Replication Strategies for a High Performance Data Grid. In *Proc. of the Int. Grid Computing Workshop*, Denver, Colorado, USA, November 2001.

[SD] 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. (Working Paper 455), McMaster University, Hamilton, Ontario, Canada (2002).

[Set03] Ankush Seth. Scalability and communication within swarm computing. 2003.

[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 Experince*, 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. University of Surrey.

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

[wsb] web site. www.swarm.org.

[wsc] web site. http://repast.sourceforge.net.

[wsd] web site. http://jaslibrary.sourceforge.net.

Diese Arbeit analysiert die Anforderungen an eine Simulationsumgebung für die Analyse der Katallaxie. Anhand von Kennzahlen wird die Auswahl der Simulationsumgebung bestimmt.

ISSN 1864-9300