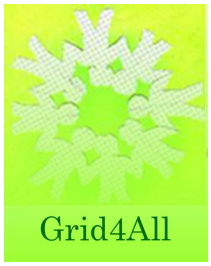

Project no. 034567

Grid4All

Specific Targeted Research Project (STREP)

Thematic Priority 2: Information Society Technologies

D6.7b Resource management for democratic grids

Due date of deliverable: 01-06-2009

Actual submission date: 25-07-2009

Start date of project: 1 June 2006 Duration: 37 months

Contributors : INRIA, FT, UPC, UPRC

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

Abbreviations used in this document

Abbreviation / acronym	Description
SOA	Service Oriented Architecture
WSDL	Web Services Description Language
DCMS	Distributed Component Management System
G4A-SIS	Grid4All Semantic Information System
ADL	Architecture Description Language

Grid4All list of participants

Role	Participant N°	Participant name	Participant short name	Country
CO	1	France Telecom	FT	FR
CR	2	Institut National de Recherche en Informatique en Automatique	INRIA	FR
CR	3	The Royal Institute of technology	KTH	SWE
CR	4	Swedish Institute of Computer Science	SICS	SWE
CR	5	Institute of Communication and Computer Systems	ICCS	GR
CR	6	University of Piraeus Research Center	UPRC	GR
CR	7	Universitat Politècnica de Catalunya	UPC	ES
CR	8	ANTARES Produccion & Distribution S.L.	ANTARES	ES

1 Executive Summary

All forms and shapes of Grids and Cloud computing technologies exist in today's world, but small organisations whether profit or non-profit still struggle with the challenge of accessing cost-efficient on-demand computing and storage infrastructures.

Consider some issues facing the following realistic scenario: a set of multi-site master course[1] jointly offered by three universities that want to jointly conduct a master's course on distributed systems and networks with objective to teach how DHTs (Distributed Hash Table) work on Internet hosts. One of the participating universities is a virtual university. Collaborative learning methodologies will be adopted and the course entails laboratory experiments where students will analyse the behaviour of different DHTs under different conditions (node leaves, joins, failures). Ideally they:

- Would like to link their minimal ICT resources to form a common networked infrastructure dedicated to the course;
- Restrict and control who use the resources; computational, storage and even content;
- Be willing to pay to obtain some more ICT resources on need, in particular when they need to execute simulation experiments;
- Would like to have collaborative tools that allow them to study and work in remote groups;
- Would like to have simple and easy ways to run their applications;
- Would like to avoid having a full-time administrator to manage this infrastructure;
- Be willing to adjust their needs according to availability of resources if this allows them to save money;
- Would like to be able to easily locate and use pertinent services that are available on the Internet.

Grid4All addresses many of these issues. It provides a new way to build and deploy democratic grids using underutilized computers on the Internet and supports users who have less professional knowledge than traditional Grid users. Such users will expect a software platform that:

- ✓ Hides the instability and volatility of the resources forming the on-demand infrastructure and provides support for modular and scalable software platforms for such environments. Application management and deployment tools simplify and improve usability for both application developers and administrators who will use such platforms to deploy their applications.
- ✓ Offers tools for different collaboration styles both synchronous and asynchronous. Collaborative applications are useful in settings where participants are remote, where often, they work in disconnected modes and resynchronize to share and discuss results.
- ✓ Ensures that such infrastructures are sustainable in the long term by providing incentives to encourage sharing. Internet grids rely on donation of resources and this works for academic projects. Incentive-based systems are necessary to render realistic, scenarios such as the above.
- ✓ Provides simple means for users and applications to locate and use resources and services most adapted to their needs. Modern applications adopt principles of Service Oriented Architectures; loosely-coupled architectures that lazily bind to the best service that matches functionally. Information and discovery systems are essential to achieve this. Semantic-based approaches are efficient but complex and finding right compromises is essential to increase adoption.

This white paper presents some key technologies developed within Grid4All to address some of the objectives[2] listed previously. It focuses on three aspects (a) application deployment management; (b) incentives and pricing of pay-by-use resources (c) means for users/software to easily locate needed services.

[1] Grid4All project members KTH and UPC participated in a joint Master program proposal submitted to the Erasmus Mundus program in April 2009. This proposal could be multi-site.

[2] Companion white papers address some of the other requirements.

2 Challenge

Cloud, Grids and Grid4All are partial models/visions/styles of distributed computing platforms. The advent of utility computing, recently called Cloud computing has reduced the complexity of allocating and using commodity hardware and software. Virtualization, pay-as-use based accounting and pricing models, software as a service, horizontal and vertical scaling technologies in applications are enabling the entry of a large number of actors in this space, but the market for cloud computing infrastructure as a service is held by a few, Amazon, Google and some hardware vendors such as IBM, Sun and HP and constitutes an oligopoly. These clouds offer a pay-per-use model with flat but opaque pricing. This oligopoly could be broken to create a competitive and efficient resource market place; organizations, enterprises (small, medium or large) and even individuals may sell capacity that they can spare and buy capacity when they have load spikes.

The large numbers of small enterprises and organisations, individual users and participants on the Internet together possess a huge amount of ICT resources. On the one hand, harnessing of these resources is done in an ad-hoc and inefficient way, and, on the other hand, in order to exploit fabric infrastructures formed using these resources, software platforms should hide complexity and mask their limitations.

Democratic grids are about deploying grids for end-users using their own collectively huge quantities of resources. Grid4All proposes a new approach to create and deploy sharing infrastructures and offers management tools and services to exploit these resources. Management costs of using such resource should not be more expensive than investing in own capacities. The Grid4All project has contributed to removing some barriers; self-management technologies that permit autonomic application reconfiguration (to heal, to tune, to optimize), deployment technologies for different types of deployment workflows and competitive discovery service for the SOA environment.

Grid4All addresses some challenges (listed below) that have been either ignored or not undertaken due to lack of compelling needs and user groups.

- There is still no easy or automated way to design, deploy and manage distributed applications.
- There is no simple way to allocate idling Internet resources and use them efficiently. Sharing models of peer-to-peer applications or Internet grids do not provide sufficient incentives such that these deployed systems are sustainable over long periods of time.
- Despite advances in SOA technologies, it is not easy to design applications as a set of loosely-coupled components that easily and efficiently exploit software available as a service on the Internet. Locating and dynamically binding to appropriate Web Services is still ad-hoc and an art.

This document analyses some key project results and sets them in perspective of democratic grid/cloud landscape.

3 Vision

3.1 Democratic grids and clouds

Democratic grids, as we conceive them are shared infrastructures operated by members of the global community and built using their own ICT resources. Community ownership of ICT infrastructure *assumes* that the community will continue to possess at least small quantities of resources and *implies* they have access to software technologies to overcome the barrier of managing and share these resources appropriately and manage applications hosted on these resources.

Cloud infrastructures are mainly single-owner data-centres; they provide services to (a) allocate and manage virtual images hosted on physical machines of the infrastructure (b) configure network addresses, both private and publicly visible (c) operate pay-per-use charging model. *Platform as service* providers may use such infrastructures to host managed applications. Generally, they cater to application classes such 3-tier web applications for e-commerce and customer relation management systems that may be monitored and scaled up or down. A downside of current clouds is lack of interoperability, growing business and marketing pressures will remedy this; applications deployed on a cloud should be able to migrate to another cloud.

These two models of computing, democratic grids and clouds, do not oppose but complement each other. Any commoditized computing resource on the Internet may be used to build clouds and democratized grids will gain by using technologies such as virtualization. This section describes a typical scenario motivating the project to show how tools developed within the project may be used to implement parts of this scenario.

3.2 An example scenario

To implement the multi-site course described in the Executive Summary, the three universities have set up an agreement to form a virtual organisation. Each university agrees to contribute some computational and storage resources to the collaboration platform.

The scenario of interest occurs when students (organised in groups) start laboratory experiments. The simulator (NS2) has libraries implementing different DHTs. Students perform series of experiments to test the behaviour of the different DHTs under different input conditions. They prepare different network topologies and input parameters (churn, traffic levels etc). When simulation terminates, students examine results using the network visualizer tool NAM. NAM requires image processing services to analyse data generated by NS2. It uses Web-Service protocols to dynamically locate and use such a service.

The simulator software is a wrapped up as a set of components with built-in management capabilities to survive volatile and on-demand environments. The architecture is made of functional and management components. Management components react to specific events such as leaves/arrivals of nodes and progress status. It follows a commonly used pattern typical in parameter sweep simulations; one computational task is executed repeatedly with different data. Input data, i.e. different model parameters is partitioned into separate files and each partition is assigned to a different task.

3.3 Management requirements we address

We use the scenario described previously to illustrate how Grid4All resource management tools and services may be employed in a democratic grid. Important aspects of the scenario are highlighted below:

1. Application Management

   Requires a way to deploy the simulator; its execution should use all possible compute nodes available within the VO. When new nodes join (login), they should be used if at least some of their

compute capacity is available. Node failures or voluntary leaves should be handled automatically and not require that students/tutors restart applications

If the amount of compute resources available in the VO is insufficient for the simulator to make sufficient progress, additional resources should be allocated. Students/tutors should not need to manually manage the application life-cycle, management components should ensure that new application components are installed, configured and activated as resources become available.

2. Locate and bind to available image processing services to visualize simulation results.

   Image processing tools are compute-intensive and often use licence based mathematical software. Such tools are often deployed as services following principles of SOA to allow applications to locate and dynamically bind to them on need. The NAM visualizer software should be able to dynamically find and use an appropriate image processing service, e.g. a service with the shortest waiting queue.

3. Utility-model

   The VO may not have sufficient ICT resources to execute all experiments within desired times; numbers of students registering cannot be completely anticipated. Experiments will be conducted by each student group and the resource quantity needed depends on the number of student groups, the difficulty of the network model and parameters that the students experiment. The course has a budget and it is cheaper to lease resources and pay-per-use than to buy additional computers.

To use the facilities that we will describe, we expect that the tools provided by Grid4All are installed and deployed. Each participating organisation assigns some compute nodes to the virtual organisation (or overlay). Each such node is installed with the Grid4All container.

- ✓ The Java based container executes on any hardware running any of the commonly used operating systems. Nodes need not be logged on to the VO all the time, the container may be shut down and nodes may join at any time. An easy to install web server serves to bootstrap joins to the virtual network overlay forming the virtual organisation.

The simulator is written using the Niche middleware and its initial deployment schema is specified with the extended Fractal ADL.

- ✓ Students use graphical tools to design deployment schemas and configure management components, e.g. set deadlines. Management components programmed by tutors provide behaviours to handle node volatility (leaves or failures) and monitor simulation progress. On detection of insufficient progress management components automatically provision new resources.

The G4A-SIS is deployed on the ICT infrastructure of one of the campuses and services are published within. Existing WSDL[3] documents may be reused along with natural language comments and descriptions to semantically annotate these; services can hence be efficiently discovered and used. The simulator queries the G4A-SIS to obtain end-point references of published services.

- ✓ The simulator is architected to exploit dynamic binding. The API offered by G4A-SIS is intuitive and can be used to locate required services. Preferences for selecting advertised services may be emitted by both providers and consumers. Simulator requests to select the least-loaded services.

The simulation program also accepts a budget. Students groups are allocated budgets by tutors, e.g. in proportion to the number of model variations they simulate.

- ✓ This budget is used when progress is unsatisfactory due to insufficiency of compute nodes.

In the following chapters we describe the main management tools and services developed in Grid4All and how these address the aspects of the scenario listed previously.

[3] Document describing Web Services

4 Paving the way to democratize Grids and Clouds

4.1 Application and service management on dynamic infrastructures

Managing distributed applications running on large-scale, dynamic environments is complex and error-prone and involves multiple interrelated activities. The process starts by finding appropriate nodes to host application elements. Correct versions of the application software must be installed on these nodes and configured. Node heterogeneity further complicates this process. Activating the application requires coordinating various initialisation actions across distributed nodes and coping with failures that occur during the process. Maintaining the application in this environment requires detecting and handling failures, which in turn requires coordinating distributed actions such as stopping, migrating or restarting application elements. Adapting to load fluctuations requires scaling up or down the set of distributed nodes used by the application. To address these challenges, tool support is essential. The following sections describe the main features of the Grid4All application deployment tools.

4.1.1 Using a component-based approach to management

Components have emerged as an effective approach to build and manage complex software systems. System elements are constructed or wrapped as components and management takes the form of inspecting and modifying the *system architecture* containing components, their interconnections and their attributes. This approach provides a high abstraction level for management, hiding idiosyncrasies of heterogeneous application elements and management techniques. For example, complex multi-tiered applications comprising diverse legacy software (e.g. web servers and databases) may be managed in a unified way. It integrates seamlessly the different phases of application lifecycle; design, deployment and operation, each modifying and producing component configurations.

The choice of component model is crucial in architectural approaches to management. Grid4All adopts the *Fractal* component model, a language-agnostic model with support for building reconfigurable systems. A key Fractal feature is the hierarchically assembling of components into composite components. Hierarchical composition reduces complexity of designing, developing and reconfiguring systems. For example, a multi-tiered application can be modelled and manipulated as a single component containing components representing each tier.

4.1.2 Separating application architecture from deployment

Deploying applications requires knowledge of both application requirements (e.g., required OS for the web server) and infrastructure capabilities (e.g., available machines and their OS). Separating the logical application architecture from its configuration on a specific physical infrastructure simplifies deployment and allows executing the same application on different infrastructures. Applying this separation, the deployment service uses architecture descriptions without infrastructure details. Descriptions are written in Fractal ADL (Architecture Description Language) and contain: (1) application structure in terms of components, their composition relationships, their interconnections, and their attributes; (2) packaging information (e.g., the names of software packages); and (3) resource requirements of components (e.g., CPU speed).

The deployment service receives as input ADL descriptions and discovers and allocates appropriate nodes to host the application components, removing need for intervention by human administrators. Resource discovery relies on low-level overlay services increasing efficiency and scalability.

4.1.3 Simplifying dynamic reconfiguration

The ADL-based deployment service is sufficient for initial application deployment but not for dynamic application reconfiguration. Dynamic reconfiguration is needed to accommodate changes in the application environment without incurring downtime. It is needed to react to resource failures, scale applications up and down as the usage fluctuates, and update applications with new software versions. Reconfiguration in Grid4All relies on the Niche[4] DCMS platform. Niche provides API to write management code that involves dynamically monitoring and controlling a particular application. Management logic is decoupled from business logic, which improves the understandability and changeability of both. Niche has a decentralised implementation and uses a structured overlay network. The current Niche prototype uses ADL-based initial deployment and low-level Java API for programmatic reconfiguration. *DepOz* provides high-level and user-friendly language support for dynamic reconfiguration.

Technically DepOz implements the Fractal component model[5] and extends it with explicit support for component packages. Remote components are accessed in location-transparent fashion and composite components can be distributed over different machines. DepOz may be used to represent both application and infrastructure elements as components; e.g. physical nodes are represented as components. DepOz offers two main capabilities. The first capability concerns navigating and querying component structures, much like XPath that is used to navigate and query XML documents. However, unlike XPath, DepOz expressions capture the dynamicity of component structures and their values are automatically updated following architecture evolutions. Using DepOz language, a query is issued once; the result of query changes as and when sub-components are added/remove. The second capability concerns defining application workflows in a concise and compositional way. DepOz provides constructs for coordinating reconfiguration e.g., synchronising the beginning of execution of components on several machines. A number of well-known workflow patterns (e.g., sequence, parallel split, and synchronisation) are supported directly and DepOZ can be easily extended to capture new patterns.

DepOZ was implemented for the Oz language as a proof-of-concept research prototype in order to provide and validate the high-level language and tool support for self-deployment and self-configuration for distributed systems and applications. DepOZ also contributes implementation of the Fractal component model to the Mozart programming environment. DepOZ can be used in the Niche platform to facilitate programming of deployment and dynamic (run-time) reconfiguration for distributed self-managing component-based services and applications developed for the Niche platform.

4.1.4 How does it compare to state of art

The problem of application deployment and dynamic reconfiguration is attracting a growing amount of attention in academia and industry. Existing systems typically use one or more of the following approaches:

(a) static software architecture descriptions that drive static deployment algorithms,

(b) workflow languages that are used to express deployment processes,

(c) constraint-based descriptions that drive constraint solving algorithms, which dynamically determine deployment configurations,

(d) AI planners that generate deployment processes.

Systems following the approach (a) are easy to use; however they do not allow developers to program dynamic reconfigurations. Approach (b) provides strong flexibility. However, unlike DepOz, existing workflow-based systems make it difficult to parameterise and compose workflows. Approaches (c) and (d) are very attractive because they allow expressing deployment requirements naturally (e.g., using constraints one can say that “there must be at least one instance of server X per 10 nodes”). However, such systems are not yet practical for large-scale distributed systems due to performance problems. Moreover, they typically require

[4] Niche is a Distributed Component Management System developed within Grid4All. It can be used to write self-managing applications (http://niche.sics.se). Niche is presented in a companion white paper.

[5] Currently implemented for the Oz language supported by the Mozart programming environment

resorting to additional mechanisms and handcrafted code in order to obtain useful results. DepOz currently uses approaches (a) and (b) providing high flexibility without sacrificing ease of use. The longer-term goal of DepoZ is to bring under a common umbrella multiple tools combining different approaches such as constraint-based approaches.

Deployment is also the focus of several standardisation efforts, including the OASIS Solution Deployment Descriptor Specification (SSD) and DMTF's Open Virtualization Format (OVF). SDD provides a standardized way to declare information about software packages and their requirements; we use descriptions written in extended Fractal ADL for this purpose. OVF targets the packaging and distribution of virtual machines. Our current implementation has no support for managing VMs but OVF will be considered when this capability is added. Other related standards are the OASIS Web Services Distributed Management (WSDM) and Web Services Business Process Execution Language (WS-BPEL). WSDM provides a standardised way to manage and monitor the status of services. The management interface provided by Fractal is richer as it includes powerful operations such as changing the configuration of composite components. WS-BPEL is a workflow language for describing business processes, but unlike DepOz, it has limited support for capturing recurring process logic.

4.1.5 Benefits

Application management relies on architectural models representing application elements, their dependencies, and their resource requirements at a high abstraction level. This rise in abstraction level enables managing heterogeneous applications in a uniform way. Moreover, it decouples applications from their underlying infrastructure, allowing the same application to operate in diverse environments, such as P2P networks and clouds. Management support in Grid4All extends from deploying to retiring the application and covers monitoring and dynamic reconfiguration to detect and adapt to changing requirements (e.g., increased demand) or underlying environments (e.g., machine or network failures). The DepOz framework, in particular, allows expressing monitoring and reconfiguration logic in a very concise and modular way. The framework includes abstractions that capture recurring management problems and solutions (e.g., deployment workflows) and can be easily extended with new abstractions.

Finally, all Grid4all management tools (e.g., deployment service, membership service, reservation service, Niche and DepOz) have themselves a componentised Fractal-based structure and can be easily adapted and extended by adding, removing, or replacing their constituent components. For example, the reservation service can be extended to allow obtaining resources from cloud providers

4.1.6 How it fits in the example scenario

The Grid4all management tools are used for creating the VO, adding students and tutors as members, and handling their authentication. The Niche platform is used to program management code that is bundled with the simulator functional code. This management code involves listening and reacting to events such as node leaving or node joining. Both types of code are deployed by the deployment service on VO nodes. Deployment relies on an ADL description that lists the simulator components (functional and management) and their deployment requirements (i.e., deploy a given component on all nodes with CPU speed larger than X). The ADL description does not specify the number or characteristics of the nodes on which the simulator is to be deployed, thus freeing users from the physical aspects of deployment. Finally, the management code ensures the automatic scaling of the simulator according to changes in the resource availability or demand.

4.1.7 Conclusions

The management tools such as Niche and DepOZ reduce the complexity of deploying, monitoring, and reconfiguring applications running on large-scale, dynamic environments. The tools include a deployment service driven by declarative application descriptions written in ADL. Although this service is easy to use, its applicability is restricted to initial application deployment. The DepOz programming framework provides

flexible support for configuration and dynamic reconfiguration. It includes support for navigating and monitoring component structures as well as for defining workflows in a concise and compositional way.

The current DepOz implementation is a first iteration and we intend to extend it in many directions, such as capturing patterns for self-managed systems as Oz libraries, and integrating with Niche in order to exploit the scalability and robustness provided by self-configuring overlays.

4.2 Connecting providers and consumers

SOA and Web Services offer a standards-based interoperability platform and allow organisations of all kinds to integrate their applications and improve accessibility to end users. To allow software and humans to find available services and ensure that services interoperate, services require visibility. Registries are about this, to publish and discover services. Services may be advertised and queried using different vocabularies even though they may mean and act the same. More so in the case of democratic grids where a large number of independent actors may publish and discover services. Registries based on simple syntactic matching offer limited understanding between providers and consumers[6] (peers), each peer having its own language. Semantic approaches to discovery where matching is based on formal descriptions of services overcome this restriction by providing a common understanding. The increase in interoperability comes with increase in development costs. Features introduced in Grid4All Semantic Information System (G4A-SIS) mitigate this.

4.2.1 Reducing complexity in negotiation-based resource allocation

Many large-scale platforms have adopted economic and market-based means to allocate resources, typically adopting bilateral interaction between interested parties, i.e. providers and consumers. Such platforms do not scale with numbers of participants, applications, resources and are complex to manage; participants need to conduct multiple sequential or simultaneous negotiations to obtain all resources. We propose a mediating platform, i.e. auction-based market-place to reduce complexity.

G4A-SIS is a major component of the Grid4All open market-place. Ontology has been designed to support atomic unitary ICT resources, and aggregations and composites. Resources are traded at formally conceptualized auction-based markets. Offers and requests describing simple, aggregate or composite resources are traded at auction-markets and published at the G4A-SIS. Applications requiring resources query the G4A-SIS to locate auctions where matching resources are traded. Matches are done based on characteristics of the ICT resources and properties of the markets where such resources are traded.

4.2.2 Simplifying the usage of rich semantic technologies

Semantic information systems improve quality of search results, but to enhance acceptance, they should (a) hide low-level details of semantic technologies from users and (b) provide means to reuse the large quantity of legacy WSDL descriptions.

G4A-SIS simplifies the job of human annotators, i.e. service developers by providing automatic methods to exploit domain ontologies, service specifications written in WSDL, descriptive information such as API documentation, license agreements and other information sources. Textual descriptions are exploited to annotate, classify and assess similarities between Web Services. In cases where lexical items in WSDL specifications are scarce, state-of-art algorithms are used to synthesize annotations.

Humans may simply provide textual annotations in the WSDL specifications through natural-language descriptions. The semantic annotation using Ontology elements is done automatically and the employed algorithms provide a high quality of annotation. The developer retains control and may validate the generated semantic specifications.

[6] Henceforth referred to as peers.

Automatic annotation also reduces errors and improves usability, i.e. improves quality of matching and discovery. Furthermore the automatic annotation designed in G4A-SIS avoids pitfalls related to phenomena such as synonym and homonym terms, typographic variations of terms, differences between granularity of services and ontological specifications. It also takes into account scarce and often misleading comments, descriptions and “tricky” names of involved service parameters being involved, as well as improper or faulty use of domain terminology.

4.2.3 Meaningful search results

When querying for services based only on functional constraints, a large number of similar services that fulfil the consumer's requirements may be retrieved. It is up to the consumer to wade through the replies before binding to a specific service. The G4A-SIS also allows selection of only those services that provide the desired functionality and that satisfy desired criteria and preferences of users.

Matched queries are ranked using both consumer and provider preferences. The Satisfaction-based Query Load Balancing (SQLB) framework of the selection component of G4A-SIS preserves both consumer and provider preferences and interests. In large-scale information systems, query allocation, i.e. the method of allocating consumer requests to providers that may satisfy the request, often privileges overall system performance, e.g. balances load across providers. The method that is used here incorporates consumer interests as well.

4.2.4 Comparing with state of art solutions

Current state of research in semantic information systems is quite mature and has released a large number of usable software products, but commercial products are not available yet. So we focus on open software issued by research. Comparison is based on the following four criteria; all compared systems provide the basic functionalities to advertise and discover resources and services.

Support for an open Grid economy:

- Simplify and reduce complexity by aiding consumers and providers to discover appropriate resource markets.

Automatic semantic annotation:

- Enhance usability by enabling reuse of existing WSDL repositories.
- Reduce complexity in development cycle while maintaining advantages of semantic discovery.

Selection and ranking of returned results:

- Allow users to distinguish among a multitude of services claiming to fulfil their needs.

API support:

- Access services in a programmatic way without needing to be aware of details of underlying semantic descriptions.

The MDS-4 from Globus Alliance and the ARC system of NorduGrid do not provide support for service discovery, essential for democratic grids. Non semantic approaches such as EGEE/ActOn and BondGrid do implement advertisement and discovery of both resources and services and even gives a ranked selection of results, but does not support the formation of Grid economies. Semantic-based systems such as InteliGrid/SRMS, K-Wf and Meteor-S do propose different levels of automated annotation, but do not propose selection of matching services by user specific preferences.

The G4A-SIS has been designed to meet these requirements and its implementation conforms to the latest standards established by Semantic Web Services technology.

4.2.5 Where does G4A-SIS fit in the scenario

The G4A-SIS is employed in items 2 and 3 listed in the section 3.3. The NAM software queries the G4A-SIS to locate and bind to the image processing service that matches functionally (service input and output types match) and with a sufficiently short waiting queue.

To lease computational resources on need, the Reservation Manager (invoked on-demand by the simulator manager) uses the services of the market-place to allocate resources. The simulator is an elastic-application and requires compute nodes. It does not mandate a fixed quantity since it can adjust to available resources. The implementation of reservation management uses services of G4A-SIS to locate auction-markets selling the needed type of compute node at unit prices not exceeding budget.

4.2.6 Benefits

Service discovery is important to allow users to locate a specific Web Service of interest from a large number of available services. Registries are essential in Grid systems to allow clients and resource brokers to discover Grid services. The G4A-SIS provides such functionality through simple Java APIs. Service providers publish their services and consumers may dynamically locate, select and access services.

The G4A-SIS is also essential to operate open resource market places. In contrast to traditional discovery systems, using G4A-SIS, providers and consumers are matched to markets where they may negotiate. Matching of individual provider offers and consumer requests is an expensive process in terms of (a) computational complexity of individual matching (b) further costs of bilateral negotiation between matched provider and consumer. This is exacerbated in democratic grids where the turnover of participants may be large. The discovery process selects markets based on resource characterization and also of economic properties such as prices.

4.2.7 Conclusions

The G4A-SIS is implemented using generic software packages with well-exposed APIs and standards-based technologies. The Pellet OWL-DL[7] reasoning engine is employed in the semantic matchmaking and the repository itself is implemented using the Jena Semantic Web Framework[8]. Jena handles RDF and OWL documents and provides gateways to data management systems; DBMS such as MySql. G4A-SIS may also be accessed using Web Services protocols; it uses the AXIS Web Services toolkit in the implementation. Annotated service descriptions are processed by the WSDL2OWL tool to generate OWL documents.

The G4A-SIS software and user documentation may be obtained from http://ai-lab-server.aegean.gr/svn/ai-lab/sis. Software prerequisites are described in detail in accompanying documentation. Specifically the following software artifacts may be downloaded:

- Source code and binaries,
- API documentation,
- User Guide and Installation guide and usage examples.

G4A-SIS extends current discovery services by permitting market-oriented characterization of resources to match offers and request. Preferences of both consumers and providers are used in this selection process. Providers and consumers may specify composite resources, i.e. bundles of multiple types of resources. This feature simplifies allocation of complex resource requests.

[7] http://pellet.owldl.com
[8] http://jena.sourceforge.com

Even though the SIS offers Java-based programmatic interfaces, this API is dependent on the underlying ontology for market descriptions. Modifications to this ontology imply changes to the API. The selection service matches consumers and providers such that their preferences are mutually satisfied. Preferences may involve dynamic parameters e.g. provider loads. Such deployments will require a scalable architecture to monitor and update preference values.

The current design has a centralized architecture; this limits scalability, restricts autonomy and breaks the desire for openness. We have finalized the design of a decentralized architecture that hopefully will remove these restrictions.

4.2.8 Exploitation of algorithms for further R&D work

The algorithm for automatic generation of a Web service's profile in OWL-S format from its WSDL description can be further exploited for research on discovery and integration of web services.

The selection service utilizes a framework for query allocation called Satisfaction-based Query Load Balancing, which can be used for distributing queries among providers in distributed settings so as to maximize the overall performance and participants' satisfaction.

4.3 Incentives to share resources in democratic grid settings

Internet computing, volunteer computing, global and desk-top computing are different terms used to designate computing on collections of heterogeneous computers scattered around the globe. In Grid computing, participating nodes are often clusters (of compute nodes), whereas in global computing, nodes are mostly individual and rarely small sets of networked computers. It bears the name '*volunteer computing*' since individual owners voluntarily donate their idling capacity to projects such as folding@home.

Peer-to-peer systems came up to enable content sharing. Free-riding, a frequent phenomenon is dissuaded through accounting mechanisms that ceil download requests as a function of upload capacity offered by the peer. Characteristics of such systems are (a) volatility and lack of stability, i.e. nodes may disconnect at will (b) heterogeneity (of nodes, communication bandwidth and latency, networks) and (c) large scale.

These systems do sufficiently encourage participation in our idea of global and democratic computing where just anybody on the network may seek ICT resources. Contrary to traditional Grids and grid virtual organisations, users of democratic grids are basically self-interested. There is a need for incentive mechanisms in democratic grids and the best candidate seems to be economically rewarding mechanisms.

4.3.1 Open market places for ICT resources

In the Grid4All architecture, resource market places execute multiple and perhaps simultaneously, many-to-many auctions that match offers and requests sent by consumers and providers. We advocate a price directed approach where consumers and producers interact at markets to allocate resources.

For our purposes, we class applications as (a) Elastic applications that tolerate variable units of allocation for a specific duration, e.g. allocation of one or more virtual machines of a specified configuration (b) Rigid allocations that mandate a bundle of one or more resource types with minimum quantities and qualities, e.g. allocation of 4 units of VM1 and 6 units of VM2 along with 100 Gb of storage. Embarrassingly parallel applications such as video transcoders and network simulators fall in the first category. They adjust to available computational capacity and their components do not communicate much. Tightly coupled applications showing complex workflows are rigid. Most of three-tier web applications such as e-commerce

are also of the second category, at least to define the resources required for the initial deployment. The Grid4All market-place caters to both classes of applications and offers:

- ✓ **A vocabulary and structure to describe ICT resources.** Resources may be combined in arbitrary configurations to represent workflows. This structure, referred to as bidding language, is available in a machine-readable and extensible XML-based format. Non-functional and quality parameters may be expressed and the bidding language can be extended to application-specific services.
- ✓ **Two auction mechanisms**, to trade resources, one for each of the two application classes cited above. The combinatorial auction allows expressions of the type "*I would like 6 units of VM1 between 10:00 and 15:00 for 2 hours each AND 100 giga of storage during the period of VM2 allocation. I am willing to pay 5 euros for this. If this is not possible, then allocate 3 units of VM3 between 11:00 and 13:00 for 1 hour each AND 4 units of VM3 between 13:00 and 15:00 for 1 hour each AND 160 Gb of storage during the period of VM2 allocation and 100 Gb of storage during the period of VM3 allocation. In the latter case I am willing to pay 6 euros*". An extended version of the well-known double auction with the k-pricing policy is suited for applications of the first category. It accepts expressions such as "*I would like at most 6 units of VM4 between 10:00 and 17:00 for at most 3 hours each. I am willing to pay at most 1 Euro for each unit of VM4 for each hour allocated.*"
- ✓ **A decentralized service to feed market information to participating traders**. Traders may subscribe for information such as aggregated (avg, min, max, total) prices of time-differentiated resources.
- ✓ **A configurable auction server** to implement auctions. Each instance of server may be deployed on-demand and be configured according to the resources that are traded, the time horizon when leasing occurs, valid bundles that may be defined over the primitive resources.
- ✓ **Centralized discovery service[9] for market participants**, i.e. consumers and providers to find appropriate markets where they may conduct price-based negotiation. For example, consumers may query the information service to find and select markets where they could buy "*6 units of VM1 between 11:00 and 18:00 where the price of one unit of VM1 does not exceed 50 cents*".

4.3.2 Pricing schemes

A pricing schema defines what the consumers pay and what the supplier gets paid. It determines how the surplus generated by the market is redistributed. They try to address multiple goals (i) be incentive compatible (ii) allow participants to derive information on the real worth of the different resources over time in order to aid them in planning their capacity needs (iii) achieve market clearing, i.e. buying prices of losing bids are inferior to the computed price and reservation price of losing offers are superior to the computed price.

Consumers (or their applications) buy resources; either to execute the complete application workflow, or to provision extra resources on-demand for deployed applications in order to improve the quality of service . For example, a three-tier application may buy additional virtual machines for its logical layer in order to improve its throughput when request rate increases. Consumers need to decide when to buy resources, the optimal duration of resources, quantities and qualities. The pricing schemes that we propose generate per-resource prices as a function of time and aids in this decision-making.

4.3.3 Market information and feedback

Resource providers strive to maximize earnings and consumers to maximize surplus, i.e. difference between the values that the service (or application) brings to the consumer and cost of leasing resource to execute them. Consumers may adjust quantity, quality, times and duration of resources by learning market prices of different resources at different times.

The Market Information System (MIS) developed in Grid4All collects information such as prices computed by auctions and routes aggregated information reliably and in a timely way to subscribers. Learning the market situation participants can refine their preferences regarding desired resource configurations and adjust them by analysis of previous and historical trade cases. MIS aids them in this decision making by providing

[9] G4A-SIS is described in detail in the section 4.2.

succinct information of prices: by category of resource and times. It provides market statistics such as volumes of supply and demand at different times of day, week, month etc. Consumers can plan capacity needs, by seeking resources at periods of low demand (low price) if convenient.

The challenge that we faced in the design of the MIS is that the market place is not centralized. Auctions may be instantiated on multiple compute nodes and at different times. Each instance operates for an interval of time and collects offers and bids from different providers and consumers. The auction clears and generates the statistics (prices, supply etc). The MIS should reliably gather all such information and report aggregated values to subscribers to aid in their decision making. Our innovation is to provide approximate information and automatically balance time and accuracy. Users define a Confidence interval and the system adjusts the number of sources that is sampled for the aggregation.

4.3.4 Comparing to state of art approaches

Despite advances in market-based approaches for Grid resource allocation, we are yet to see commercially operated resource and cloud market places. Commercial cloud offers such as Amazon EC2, Sun Grid Compute Utility propose pay-per-use model. Prices are fixed per hour of VM usage and the pricing scheme is opaque. Amazon has met with a great success and its consumer base is increasing, but we do know the percentage of successful allocation requests[10]. Prices are not differentiated according to the performance of virtual machines and at this time advanced reservation of resources is not possible. The Sun Grid utility facility is similar; it charges flat rates for each hour of consumed CPU. Other restrictions cited for the Amazon offer applies to Sun as well.

GRIA [www.gria.org] is an economic middleware and focuses on guaranteeing service level agreements. Prices are fixed in GRIA. The EGEE [www.eu-egee.org] project operates a large computing infrastructure for the research community. It incorporates a Price authority that assigns prices to Grid resources. Prices may be assigned either manually or by a default algorithm that increases prices for jobs that have requested lower waiting times.

GRACE [www.gridbus.org] recommends architecture and provides a set of services to implement an Economy Grid that regulates supply and demand. It has a centralized architecture with a registry to store provider advertisements. Brokers select appropriate resources to execute jobs so as to maximize utility. The main parameters used by the brokers are the deadline and price. A centralized accounting and payment service complete the platform.

Compared to these systems, we propose a loosely-coupled architecture that recognizes that multiple pricing and allocation schemes need to be supported. Our combinatorial auction mechanism supports allocation of composite (bundled) resources. Our market information service maintains price histories. These may be analyzed for temporal and spatial patterns. Our market-place architecture is not tightly coupled to execution management in Grid nodes and behavior of their resource management systems.

More recently GridEcon and Sorma FP6 projects propose systematic approaches to construct grid and cloud market places. We will be examining in greater details the results of these projects to further enrich our market place services and tools.

4.3.5 How fits in the example scenario

The Grid4All market place offers a utility-allocation feature for democratic grids (item 3 in section 3.3). In our scenario, using the market-place services, compute nodes can be allocated when progress of simulation is insufficient. The simulator's management component monitors the simulation progress and when needed requests the VO Reservation Manager to lease resources at the market place. The Reservation Manager hides the complexity of negotiating at the market; as input, it is given (a) description of resource characteristics (quality and quantity) and the earliest starting and latest ending times for which resources are

[10] If currently demand exceeds supply and what policies are used when demand exceeds supply

required and (b) budget. The implementation uses APIs of the different market place services. As a result, the Reservation Management obtains an Agreement object (if it has succeeded in allocation), that allows resources leased at the market place to be accessed and used in simulation.

4.3.6 Benefits

Configurability of the auction server is important as this permits specialization of specific components such as auction pricing rules without needing complete re-implementation. The architecture-based approach facilitates deployment, the rich bidding language supports a wide variety of market mechanisms and each instance may be configured to specialize in characteristics of traded resources, time constraints and other attributes such as location.

The combinatorial auction mechanism responds to multiple criteria. First, it computes market-clearing commodity prices of resources as a function of time. This feature is useful for participants to not only understand why they have lost but enable them to compute prices for future requests. Mathematical tools may be applied to find patterns on historical prices to help in decision-making. Second, some applications may require mandatory bundles of composite resources. Combinatorial auctions avoid the exposure problem. Third, workflow-based applications are supported to a large level.

The market information service is an important tool to understand how the market behaves and evolves. By aggregating and providing summarized information of the overall resource market place, this service allows participants to adjust their requests and offers and maximize their utility. Global and consistent information is expensive to obtain and not even necessary. The MIS allows subscribers to control the level of confidence they expect and it automatically adjusts the precision of information.

Overall the architecture of market place allocation that we propose is loosely coupled and does not express rigid requirements on execution management and provider node resource management.

4.3.7 Conclusions and Future work

We expect to see evolutions in Cloud computing and new business models to arise. In the long run, the Cloud market will expand. Interoperability will break vendor lock-in and enable a large number of actors to participate. Entities may act as market-place operators or brokers and offer value-added services, even though clearly there is room for future work before this vision. Some avenues for future work are:

- Intelligent and automatic tools to aid consumers and providers in deciding what resources they need, when they need them, for how long and how to price them. These tools should take into account the application performance model, user specific utility functions and feedback from the market. These could be termed as capacity planning tools that help decide when it is advantageous to sell resources and when to buy resources.
- Platform integration of technical and business aspects such as trust, security, accounting and payments. In the context of cloud computing, specific services to manage virtual machine (images), to analyze application performances are additional services that add value.
- Evolution of technologies and how participants may learn such evolutions.
- Software engineering methodologies and tools to enable different autonomous participants with different local knowledge to adapt to evolutions in market pricing and negotiation mechanisms.

4.3.8 Exploitation of algorithms for future R&D work

The pricing mechanisms and the heuristics-based algorithms employed to resolve the allocation problem in the combinatorial auction bring trustable and useful information to participants, but nevertheless have some limitations. Commodity-market clearing prices or their close approximations are computed using valuations

and prices set on bundles, both over quantities and over durations. In some cases, we may not be able to find acceptable prices as a function of time; this requires improvements on the proposed scheme such that prices can be correctly interpreted.

An open market-place may employ a wide category of pricing-based or even other multi-criteria based negotiation protocols and mechanisms. Actors in the market place may take different roles, e.g. buyers, sellers, auctioneers. These agents may not be doted with all possible negotiation capabilities, either for economic purposes or for interaction-behaviour purposes. We need to minimally allow participants to understand their capability with respect to a given auction protocol and possibly allow the participant to adapt to behaviours of the auctioneer. We advocate an open service oriented market place where negotiation protocols may be formalized using choreography description languages such as WS-CDL. It is interesting to pursue work in this direction to detect compatibility, i.e. is a trader doted with the behaviour expected of the auction at which it wants to participate and then to find mechanisms that permit traders to adapt.

5 Conclusion

Distributed computing is currently dominated by large data centres that operate by economies of scale. These centres are operated by providers such as Amazon who leverage experience they have acquired in managing and optimizing clusters. Today's data centre networks rely on virtualisation technologies and load-balancing hardware and software, complemented by advanced capacity planning and optimizing techniques, which allow them to increase resource utilisation and to offer, compute services at competitive pay-as-use prices. While infrastructure-as-a-service cloud products are agnostic to programming models and basically provide clients with access to virtual machines, storage and network capacity, platform-as-a-service products provide APIs, programming models, development environment, application administration and varied levels of platform services (authentication, authorization, billing, databases etc).

Cloud computing is dominated by a few actors who capture the essential of the market. To sustain growth at this rate the possibilities are (a) continuing with the same trend with still larger data centres forming within themselves networks that can be compared to the Internet itself, (b) using larger numbers of data centres, where consumers and large companies resorting partially to cloud computing services may themselves sell their spare resources, and finally (c) reaching the vision of world-wide cloud computing where any actor large or small may buy or sell resources at resource market places.

This paper has described solutions for three main challenges for distributed computing identified as part of the Grid4All project. These challenges and their solutions are highly relevant to all the three aforementioned trends in cloud computing.

- **Flexible application management:** Cloud platforms offer a wide range of services and tools for application developers and administrators. These services frequently include application management in the form of deploying applications, scaling applications up/down or tolerating failures. Nevertheless, most cloud platforms specifically target one or two widely used application domains and architectures (e.g., Google AppEngine targets 3-tier web applications). The management tools that we propose are widely applicable, not restricted to specific application domains. Importantly, the tools are explicitly designed to capture and support recurring application patterns; they can thus serve as the basis for domain-specific cloud platforms.
- **Promoting efficient allocation of resources:** Cloud operators sell their services using fixed prices. How exactly prices are determined by these operators is not known, but they rely on estimation and prediction of demands and knowledge of their own operating and provisioning costs to establish prices. These prices, while sufficiently low to attract new customers, may prove to be too expensive on the large run, if a large category of consumers where to completely outsource their IT requirements. We believe that market place technology enables the participation of a large number of providers (who may themselves be consumers another day). Competitive pricing that takes into account fluctuations in supply and demand will keep consumers happy by lowering costs and keeping vendors with satisfactory revenues.
- **Improving quality of interoperability:** Service Oriented Architectures promote loosely coupled, highly interoperable and dynamically composable applications and services; integration of distributed components improves efficiency and time-to-market in deployment of new applications. A large number of services exist in IT environments and this trend will continue. Discovery and selection of the best services conforming to user's needs that resolves heterogeneity in services capability and interfaces are best addressed by automation using semantic technologies. However if gain in efficiency is offset by difficulty of use developers and users will not adopt these technologies. The G4A-SIS contributes to democratisation of semantic technologies by helping to reduce their complexity.

Level of confidentiality and dissemination

By default, each document created within Grid4All is © Grid4All Consortium Members and should be considered confidential. Corresponding legal mentions are included in the document templates and should not be removed, unless a more restricted copyright applies (e.g. at subproject level, organisation level etc.).

In the Grid4All Description of Work (DoW), and in the future yearly updates of the 18-months implementation plan, all deliverables listed in section 7.7 have a specific dissemination level. This dissemination level shall be mentioned in the document (a specific section for this is included in the template, both on the cover page and in the footer of each page).

The dissemination level can be defined for each document using one of the following codes:

PU = Public

PP = Restricted to other programme participants (including the EC services);

RE = Restricted to a group specified by the Consortium (including the EC services);

CO = Confidential, only for members of the Consortium (including the EC services).

INT = Internal, only for members of the Consortium (excluding the EC services).
This level typically applies to internal working documents, meeting minutes etc., and cannot be used for contractual project deliverables.

It is possible to create later a public version of (part of) a restricted document, under the condition that the owners of the restricted document agree collectively in writing to release this public version. In this case, a new document code should be given so as to distinguish between the different versions.