ICENI: An Open Grid Service Architecture Implemented with Jini

Nathalie Furmento, William Lee, Anthony Mayer,
Steven Newhouse, and John Darlington

London e-Science Centre
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
lesc@ic.ac.uk — http://www.lesc.ic.ac.uk

Abstract. The move towards Service Grids, where services are composed to meet the requirements of a user community within constraints specified by the resource provider, present many challenges to service provision and description. To support our research activities in the autonomous composition of services to form a Semantic Service Grid we describe the adoption within ICENI of web services to enable interoperability with the recently proposed Open Grid Services Architecture.
Keywords: Computational Grids, Web Services, Semantic Grid

1 Introduction

The early Computational Grids, federations of distributed resources, demonstrated the ability to securely access remote computational resources through a single 'Grid' identity [1, 3]. The recent proliferation of computational resources and their adoption by the applied science community has led to a new user community – e-scientists. The functionality provided by these Grid infrastructures has expanded in recent years to support the challenges presented by the distributed petabyte data collections contained within the Data Grids inherent in many of today's scientific experiments [12, 9].

Recently, the demands from the e-commerce and e-science communities have forced the convergence of technology between these two fields. Within the e-science community, it has become clear that the underlying infrastructures need to be portable, easy to deploy and straightforward to extend. While both communities have been faced with a proliferation of services from the different organisations. Finding the 'best' service capable of meeting the needs of a user, or a community of users, is inherently complex either the service is a stock quote or the solution of a partial differential equation.

It is therefore essential that services be transparently and efficiently composable to meet the needs of the end-users. To this end, two aspects of the service need to be defined: its functional characteristics encapsulated within a well-defined and discoverable interface, while an appropriate meta-data schema describes its non-functional

0-7695-1524-X/02 $17.00 (c) 2002 IEEE

behaviour. By developing infrastructures that can comprehend these meta-data ontologies, services can be automatically composed to meet the defined needs of both the user and the resource providers. We classify such an infrastructure as a Semantic Service Grid - where services may be autonomously composed to meet the defined needs of the user.

In [13], Tim Berners-Lee defines the Semantic Web as 'an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.' Likewise, the Semantic Service Grid can be described as an 'extension of the current Grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation'.

This paper describes the evolution of ICENI (Imperial College e-Science Networked Infrastructure) to meet the demands of a Semantic Service Grid. We describe how ICENI is being extended to enable integration with other infrastructures built upon web-service protocols such as the recently proposed Open Grid Service Architecture (OGSA) [4]. We feel that Semantic Service Grids need to be built upon four key concepts:

- All services are annotated with information relating to their capability;
- All services are provided for a specified duration and community - expressed through a 'Service Level Agreement';
- New services, with a new Service Level Agreement, are instantiated through negotiations with an existing service;
- Services within such an architecture are presented by different views.

2 The Emerging Service Grid

The key requirement in any service oriented Grid middleware is a description of its interface and capability. This information is essential in order to compose a set of services to meet a user's requirements. Very few of the existing Grid infrastructures (e.g. Globus [3] and Legion [10]) provide the means to access this information for a generic interface. Instead they rely on the definition of several 'well-known' protocols accessed through a client side API. Our work over the last two years has been focused on the development of ICENI, a Grid middleware designed to contain and express such meta-data.

2.1 ICENI

Within ICENI, we make a clear separation between the management of a resource and its use by a wider community [6, 7]. All computational, software, hardware or storage resources appear as Jini services within a private administrative domain. The resources within the private administrative domain are published as services in a public computational community by the domain manager, which annotates each resource with information relating to its access, usage policy and duration. The services within a computational community are accessed by the users and higher-level services through well-defined interfaces and an XML based request / response protocol based around the exchange of documents defined by well known schemas.

2.2 Web Services

Web service applications have recently been popularised through e-commerce adoption and strong industrial backing. The development of web services protocols such as the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and the Universal Description, Discovery and Integration (UDDI) protocol has simplified the integration and interaction between services provided by different e-commerce organisations. These three protocols provide the means to discover services, their capability and to interact with them in a platform neutral mechanism.

The requirements from the developers of Grid middleware coupled with the emerging capability of web service's technology has created a new avenue for Grid service providers which allows them to benefit from, and contribute to, the e-commerce community. However, existing web services protocols have not been designed for the use in compute or data intensive infrastructures although optimised implementations of the SOAP protocols have been produced [2, 11].

The dynamic nature of services in the Grid compared to the relatively static provision of services in e-commerce environments means management and searching for available resources become increasingly complex and unscalable. These limitations of web services protocols within the Grid are well recognised. ICENI, alongside other Grid middleware, recognises the suitability of web services as high-level protocols for service co-ordination. These protocols may be used to discover and select the most optimal low-level communication mechanism thereby achieving inter-operability and high performance within a specified quality of service.

2.3 Open Grid Services Architecture

The recently proposed Open Grid Services Architecture (OGSA) [4] has brought about a convergence of the grid and web services communities. It leverages commercially supported web services protocols to build a Grid infrastructure. OGSA adopts the general web service approach for describing interface and behaviours of all Grid services by using WSDL. Extensions to WSDL are proposed within the OGSA to address the previously defined limitations of web services and to describe service versioning, compatibility, binding-neutral communication and service state. A technology preview of the OGSA reference implementation developed by the Globus team has been recently made public. The ICENI container extends the reference implementation in order to closely track the changes in the proposed specification.

3 The Semantic Service Grid

The Service Grids outlined in the previous section provides a basic infrastructure for building other, higher-level, application oriented services. If these services are annotated with meta-data relating to their capability and use, then users, or agents acting on their behalf, are able to autonomously discover, understand and exploit this service related meta-data to meet their own defined goals or requirements. The development of such infrastructures, which we class as a Semantic Service Grid, is essential if they are

to transparently meet the needs of the applied e-scientist. As a move towards such an architecture, we are continuing to develop ICENI by exposing our existing services and meta-data structures within a web services framework.

3.1 ICENI Service Architecture

The ICENI architecture is constructed from a clear abstraction between resources and their exposure as user accessible services. The technical capabilities of the resource, its attributes, are expressed within an XML document by the resource administrator through a defined XML schema. Interaction with a service is also through XML documents based upon a specified schema using a request / response mode of interaction. All services are derived from a resource and are exposed with a defined 'Service Level Agreement' (SLA) or contract, that specifies how and by who the service the resource may be used.

Inherent to this model is the ability for another service or user to negotiate and instantiate a service with a new SLA. This new service may only be accessible to a particular subset of the original service's user community or its functionality.

A basic service interface, common to all services within our architecture, is defined using WSDL. By building upon the recently announced OGSA, we will enable services to:

- Discover the supported protocol schemas and versions;
- Discover the supported attribute schemas and versions;
- Extract the defined attributes;
- Interact with a service through its supported service protocols;
- Negotiate the creation of a new service with a new service level agreement;
- Accounting (charging) information.

By versioning the attributes and protocol languages through XML schemas, we are able to discover services that we can directly or indirectly (through translation) 'understand'.

3.2 ICENI Web Services

By integrating ICENI with the OGSA infrastructure, we utilise web service protocols as a mean to provide an interoperable layer between other grid middleware. ICENI employs Jini as the primary service infrastructure to take advantage of dynamic registration, service lookup, distributed object access and the platform-portability provided by Java. We acknowledge a tight coupling to the Jini technology limits the interoperability of the system but web services on their own do not cater for the dynamics of the grid environment. We believe a two-layer architecture utilising Jini at the community level and web services at the global layer can realise the benefits and flexibilities of the two technologies. Grid implementation compatible at the Jini layer can communicate directly, while disparate systems can resolve to the web service layer. The ability to select transport-level binding in web services would provide further scope for developing high-performance transport for data-intensive grid application.

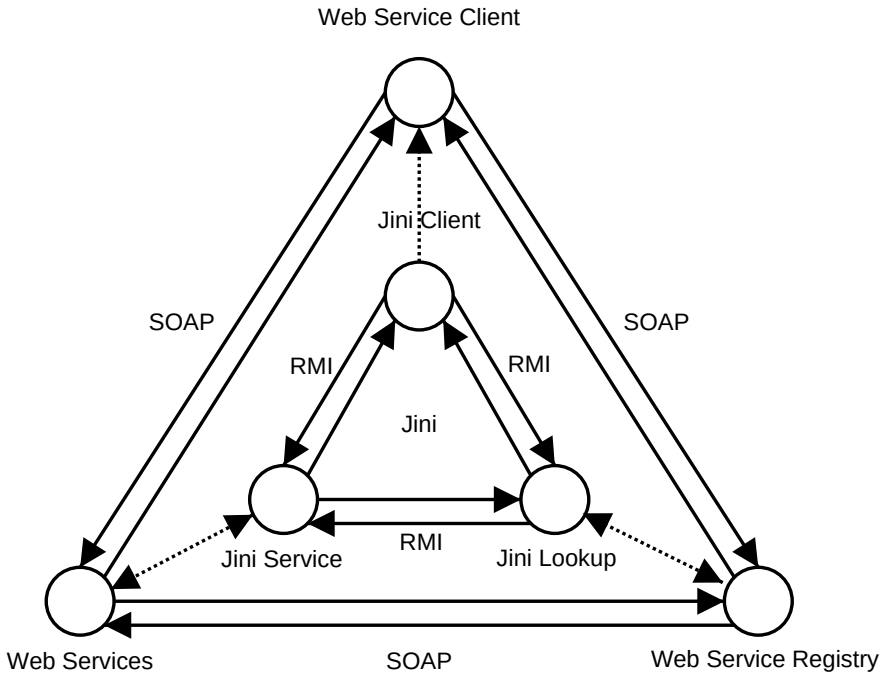

Fig. 1. Jini and Web Services.

ICENI bridges the two complementary technologies by introducing a web service container as the hub to expose Jini object as web-enabled grid service. We recognise that it is impractical for each transient Jini resource to be an OGSA-compliant container. As a result, ICENI maintains a persistent set of containers be served as islands for services in the community to dock. These Jini services bind to the containers in order to become OGSA-capable. The container acts as a SOAP-to-Java endpoint, a security gatekeeper and a set of service registries defining community boundaries.

Our implementation is based on the GSI-enabled web service container developed by the Globus team as part of the OGSA reference implementation [4]. We have enhanced the implementation by exposing it as Jini object to take advantage of the distributed event model and leasing mechanism. A resource modelled as Jini object can explicitly lookup the container and deploy itself as web-enabled grid service. The Jini distributed event model leverages the problem of garbage collecting the states of transient services. The container receives registration events posted by the Jini lookup service to maintain a consistent service availability in the registry.

The deployment agent is introduced as an optional translation layer between Jini services and the exposed OGSA-compliant grid services. The deployment agents, who are themselves Jini object, monitor a set of Jini lookup services for the arrival and departure of services and containers. Upon arrival of a Jini services understood by the deployment agent, it dynamically generates stub that is deployable to the particular

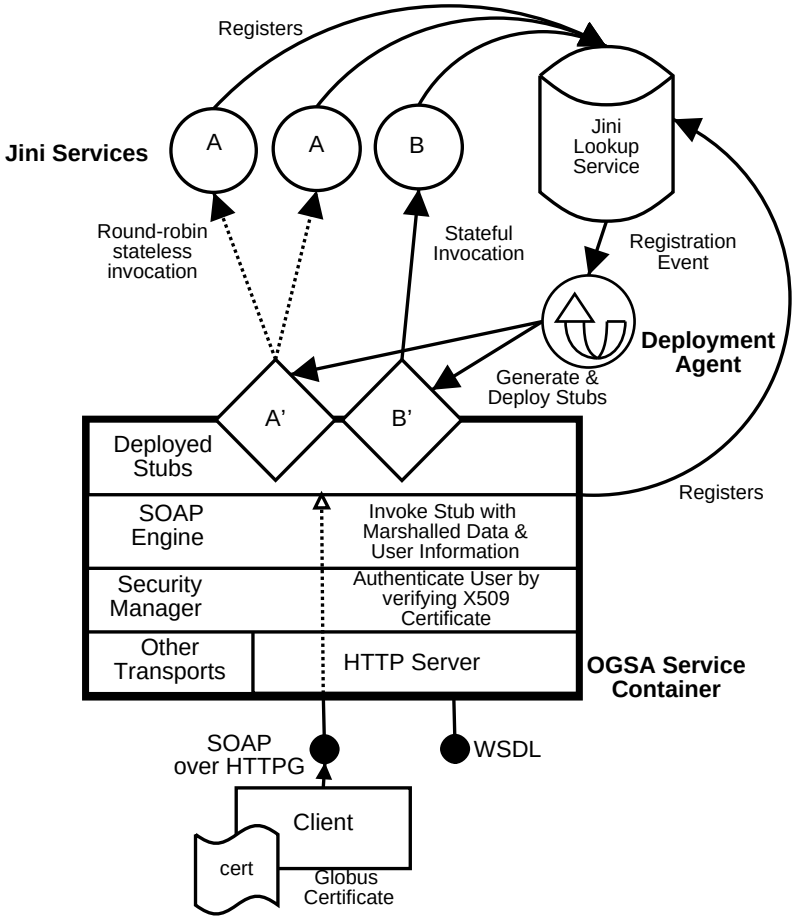

Fig. 2. Web Services Container - OGSA & Jini Integration

type of containers known to the agent and the corresponding WSDL document which describes the exposed methods of the grid service.

The advantage of this translation is twofold. It allows us to provide fine-grain control of service promotion across virtual organisation based on defined policies. By virtualising the underlying service, different views of the same object can be exposed, such as the set of supported methods for a particular community served by the given container or different security constraint. Java's dynamic class generation and loading mechanisms enable a dynamic and transparent translation process. It enables us to experiment with various invocation strategies by wrapping method call in the stub, such as stateless round-robin invocation to a group of Jini services of the same type or stateful invocation to a persistent Jini object.

Service Discovery. To facilitate dynamic service discovery, client must be able to query available services through a registry. In a Jini environment, application queries the Jini lookup services by matching data members of an Entry object associated with the services. ICENI has implemented the OGSA Registry port type which wraps the Jini lookup mechanism for service discovery. The Jini leasing mechanism ensure transient services will be removed from the registry when they cease to exist.

The standard OGSA GridService port type mandates a set of methods for querying service data. This mechanism allows us to expose ICENI service specific contract information as well as the comprehensive XML description of the ICENI components [6]. We envisage the development of a rich query interface (e.g. XQuery, XPath expression, etc.) be used in the registry for querying the rich set of metadata associated with the service.

Each grid services deployed on the container has an associated OGSA Grid Service Handle (GSH). A handle mapper service provides the translation between GSH to a Grid Service Reference (GSR). Client can then use this GSH to locate the Grid Service Reference (GSR) encoded as WSDL as the network-wide pointer to the specific grid instance. OGSA does not mandate the GSR strictly be a WSDL document. It is therefore possible to encode the GSR as a Java stub which the client can use to directly invoke the underlying Java Jini instances. This indirection provides an avenue for implementing load-balancing and fail-over scheme with multiple instances of the container work in cooperation to serve a community.

Authentication and Authorisation. An open system incurs a high security risk. The exposure of valuable resources through the web service container needs to be secured through effective authentication and authorisation. ICENI uses the X.509 certificate as the authentication mechanism. The container adopts the Grid Security Infrastructure [5] using SSL authentication and encryption layers as the underlying transport for the SOAP endpoints. Clients identify themselves using their certificates during the SSL handshake. The container extracts the distinguished name from the client certificate and passes the distinguished name and service ID pair to the Identity Manager to authorise access based on the corresponding service contract.

3.3 Service Factory

Within ICENI, we expose a resource managed from a private administrative domain to a community of users through a SLA or contract. This agreement specifies the individuals, groups and organisations that are able to use the resources provided by that service and the costs (if any) occurred in their use. An example of such an agreement is given in Appendix A. Any interaction with the service has to meet these access controls and usage requirements. A member of this community may negotiate ‘exclusive’ access to a subset of these resources for their own use. This exclusive access is represented by the creation of a new service (associated to the parent service) for a specified duration with a SLA requested by the community member.

Therefore the basic service interface provides a method that allows the service to be spawned for a particular set of users and duration specified by a contract. The new

service will have the access restrictions specified by the contract and will appear as a new Jini service and therefore as a web service instance.

The new contract has to be a sub-contract of the existing contract. Meaning this mechanism cannot be used to extend the access's capabilities of the service, but should be used to make the service available to a smaller set of users or for a shorter period of time.

As an example, consider the use of a computational resource consisting of 16 processors by a project's Principal Investigator (PI). The computational resource is instantiated as a service defined within the SLA as being accessible to a particular group of local users, including the PI. From this service, the PI is able to negotiate a subset of these resources for their own use and that of their local collaborators. For instance, 8 processors may be made available for the exclusive use of the PI and his collaborators for a period of 5 hours for which the PI will be invoiced for their use. The PI and his collaborators are then able to utilise the computational service for its duration after which it is reclaimed.

We feel that this generic model may be applied to the basic resources within a Grid environment such as storage, networking, data sets, applications, user accounts, etc.

4 Discussion

ICENI entities can thus be considered both as OGSA compliant web services (as the specification is finalised), and as Jini objects within a local Jini space. There is no redundancy in this pattern; rather the ICENI resources present a different face according to the role in which they are being used. This enables a notion of 'views' – different actors will communicate with and influence ICENI entities according to their particular needs. The public Computational Community view presents the ICENI resources as web services which are accessible by the world at large, while the private Administrative Domain view allows entity examination through a Jini mechanism. The notion of views need not be restricted to the synergy between the Jini and Web Services protocols however; it can be envisaged that other views may be specifically tailored to support end users steering applications and manipulating entities in terms of dataflow, or for software developers to design application components that interact with ICENI resources, utilising tools capturing control flow. These aspects of ICENI are discussed in other papers (for example [6, 8, 7]), and demonstrate the potential for a completely integrated grid programming environment with multiple points of view.

5 Conclusions

Our current work with ICENI has demonstrated the use of Jini as a dynamic service management framework within a Grid environment. However, we recognise that the adoption of Jini restricts inter-operability as both the client and server endpoints need to to be implemented in Java. By building upon standard web service protocols, we are able to promote inter-operability and to provide a structured mechanism, through WSDL's extensibility element, to access a service's functionality and meta-data.

These extensions include a new binding specification for expressing Jini endpoints, a description for binding-specific security mechanisms and references to the rich service meta-data. We envisage the WSDL document will serve as a network-wide pointer to the deployed service. Higher-level mechanisms can exploit its self-describing nature to support the autonomous service composition to meet the needs of users and dynamically adapt to the underlying availability of resources. The proliferation of services within a service grid environment is simplified through the use of views.

We have shown how the powerful mechanisms within ICENI for resource sharing through service level agreements can be exposed as web services. Our web service infrastructure will track the OGSA as it develops and will ultimately result in an OGSA compliant interface to the Jini encapsulated resources. This work was demonstrated in April 2002 at the opening of the UK e-Science Core programme's National e-Science Centre (NeSC) in Edinburgh, Scotland.

Acknowledgement

This work is funded by the DTI/EPSRC e-Science Core Programme
THBB/C/008/00023 and by EPSRC grants GR/R74505/01 and GR/R67699/01. The equipments used in this work was funded through HEFCE JREI awards GR/R04034/01 and GR/M92355/01.

References

1. T. DeFanti, I. Foster, M. Papka, R Stevens, and T Kuhfuss. Overview of the I-Way: Wide area visual supercomputing. *International Journal of Supercomputing Applications and High Performance Computing*, 10(2):123–131, 1996.
2. D. Gannon *et al*. Extreme Computing Lab. `http://www.extreme.indiana.edu/xgws/`.
3. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
4. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Available at `http://www.globus.org/research/papers/ogsa.pdf`.
5. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *ACM Conference on Computer and Communications Security Conference*, pages 83–92, 1998.
6. N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington. ICENI: Optimisation of Component Applications within a Grid Environment. Under review with the Journal of Parallel Computing.
7. N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington. An Integrated Grid Environment for Component Applications. In *2nd International Workshop on Grid Computing, Grid 2001*, volume 2242 of *Lecture Notes in Computer Science*, Denver, USA, November 2001.
8. N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington. Optimisation of Component-based Applications within a Grid Environment. In *SuperComputing 2001*, Denver, USA, November 2001.

9. Grid Physics Network. http://www.griphyn.org/.
10. A. S. Grimshaw and W. A. Wulf *et al.* The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1):39–45, January 1997.
11. Ninf Project. Ninf: A Global Computing Infrastructure. http://ninf.apgrid.org.
12. The DataGrid Project. http://www.eu-datagrid.org/.
13. W3C. Semantic Web. http://www.w3.org/2001/sw/.

A Example of a XML document describing a Service Level Agreement

```
<?xml version="1.0" encoding="UTF-8"?>

<contract
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.iceni.ac.uk/xmlSchema/contract"
    xsi:schemaLocation="http://www.iceni.ac.uk/xmlSchema/contract
                        /vol/iceni/icpc-core/data/xml/grid/contract/CONTRACT.xsd"
    duration="1200">

    <allow>
        <entity type="person" name="CN="JohnDoe,OU=doc.ic.ac.uk,O=LeSGrid,C=UK"/>
    </allow>

    <deny startDay="friday" startHour="13" stopDay="sunday">
        <entity type="group" name="ICPC"/>
    </deny>

    <allow stopDay="saturday" stopHour="20">
        <entity type="organisation" name="EUDataGrid"/>
    </allow>

</contract>
```