Licensing the Use of Grid Services

Kay Dörnemann[1,2] and Bernd Freisleben[2]

[1] Information Systems Institute, University of Siegen,
Hölderlinstr. 3, D-57076 Siegen, Germany
[2] Department of Mathematics and Computer Science, University of Marburg,
Hans-Meerwein-Str. 3, D-35037 Marburg, Germany
email: {doernemk, freisleb}@informatik.uni-marburg.de

Abstract

In this paper, a flexible approach to license the use of WSRF-compliant Grid services as implemented in the Globus Toolkit 4 is presented. A license definition and recombination language which allows to create new licenses on demand in a fine-grained and user dependent manner is introduced. Implementation issues for some components of the proposed licensing system are described.

1 Introduction

To prevent software piracy in service-oriented Grid computing environments, the deployment and use of Grid services must be protected. There are several possibilities to achieve this goal:

- Grid services based on the Web Service Resource Framework (WSRF) standard and ordinary web services are accessed through a standardized interface and hosted by a service provider in a service container. It is possible to limit the access to such a container through firewalls and other transport layer techniques, but in general the granularity provided by this solutions is too coarse.
- Application layer protocol authentication mechanisms, SOAP-Header extensions or WS-Security can be used to authenticate a user and forbid access to a service. With these solutions, it is not possible to specify precise access rules or to account for the use of a service. Access can only be prohibited for users, machines (i.e. by IP addresses) or networks.

In this paper, an approach to license the use of Grid services in a fine-grained manner is presented. It is based on WSRF Grid services as implemented in the Globus Toolkit 4. D'Andrea et al. [1] have already pointed out the importance of developing appropriate licensing schemes for web services. Since Grid service operations often have a higher time exposure, consume more CPU cycles, memory and bandwidth than web services, licensing their use is even more important, particularly if service-oriented Grid computing should make its way out of the scientific laboratories into the business world.

Both service providers and service developers can benefit from our proposal. For example, service developers can use third party license systems like flexLM to protect their services. Such solutions require that the source code is available to insert the license protection code. Normally, the source code is not available for service providers.

This makes it hard to protect their services in such systems. The approach we are presenting enables license protection for all deployed services without any needs to change the service code. Our prototype implementation is an extension of Apache Axis and the Globus Toolkit 4.1.

The paper is organized as follows. Section 2 presents the proposed approach to licensing the use of Grid services. Section 3 introduces the proposed license specification language. Section 4 describes implementation details. Section 5 discusses related work. Section 6 concludes the paper and outlines areas for future work.

2 Licensing Grid Service Use

An overview of the proposed system for licensing the use of Grid services is presented in figure 1. The main components are the gatekeepers on the client- and server-side, the manager service with several related components and one or more optional services. The tasks of the server-side gatekeeper is to manage and monitor the connections to the different services. The client-side gatekeeper is an optional component that can be used to authenticate clients which do not use web service authentication mechanisms. Both gatekeepers perform their duties transparently to the established connections to the services. Thus, it is not necessary to urge the client or service code to enable protection. The third main component is the manager service. This service provides a Grid service interface to obtain and manage licenses for users and administrators. The other components shown in figure 1 are part of the deployment of the manager service. They are designed as standalone components and can therefore be used by other service implementations.

A typical application flow using the presented licensing system could be as follows. The administrator chooses a license and links it to a service using the administration client. This information is stored in the Web Service Registry. Let us assume that the administrator has selected a time based license that allows access for users from his/her network during day time and access for all users at night. Now suppose that during the day a user from the administrator's network connects to the Grid service using the client that belongs to it. The server-side gatekeeper fetches the license information for this service from the Web Service Registry. The access is now granted because the client's IP address stems from the network that is allowed to connect at daytime. The client can now use the service without taking notice of the licensing system. An user from another network who tries to connect to the administrators network at daytime gets an exception, telling that the service is not available at this moment.

As shown in figure 2, the license manager service is split into several components, interfacing with the server-side gatekeeper. The manager in this figure actually is the web service interface described above. The scheduler keeps track of limited (by time or other limitations) licenses and triggers the server-side gatekeeper when a license expires or a license starts. The second component is the Web Service (WS) Registry. It stores information about which type of license is needed to use a service, resource, container or a method. Another component is the "In Use Licenses Registry" which keeps track of all licenses and services currently in use. The monitoring unit observes the connections for accounting and billing purposes but also for further analysis and

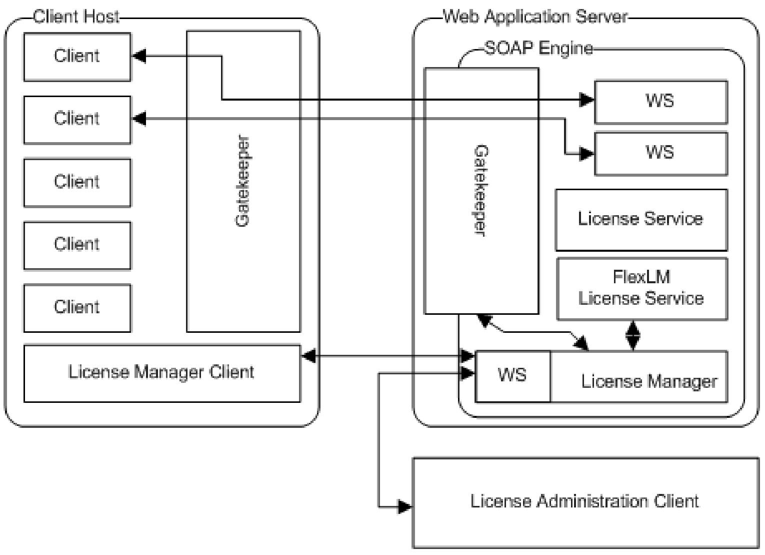

Figure 1: Overview of the system for licensing the use of Grid services

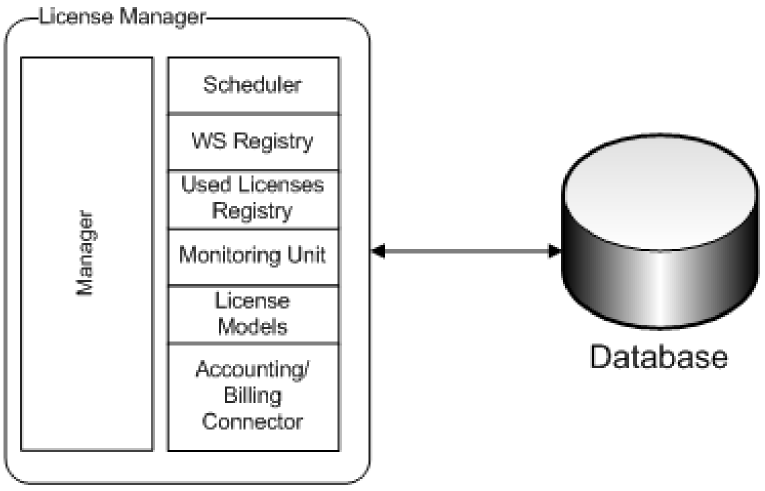

Figure 2: License Manager Service

surveillance. It cooperates with the Accounting and Billing Connector which provides an interface for accounting and billing software. The License Models component is a database connector that provides interfaces to create, edit and store licenses for general or service-related licensing.

There are three main kinds of licenses: basic licenses provided by our system, licenses provided by developers and licenses based on third party license management systems (like flexLM from Macrovision). By providing a range of basic licenses and a license recombination and description language, our system allows to create fine granular licenses as described in section 3. This language allows the user to recombine and define new licenses. Third party license systems are represented by optional Grid services or plug-ins, each with a specific interface. These interfaces allow to gather information about licenses and to obtain a license. The MDS Connector is an optional component that registers information about licenses in the Globus Toolkit 4.1 MDS 4. All components use a relational database for persistent storage of information.

The system comes with a manager and an administration client. The first one enables the user to order and manage obtained licenses, and the second one is the administration client. Its purpose is administration and server-side management of licenses, such as creating licenses, removing licenses, reconfiguring licenses and binding licenses to methods, resources, services or containers. Both clients provide a graphical user interface and an application programing interface that can be used to integrate the clients into other applications. The graphical user interface of the administration client allows to drag and drop licenses and edit their properties to create or recombine licenses. Both clients can be used as a standalone application or as an Eclipse plug-in. Both clients provide collaborative functions to get remote help from an expert. In the case of the managing client, this expert could be the administrator or the service provider. In the other case, a developer or software distributer could be the expert.

3 License Recombination and Description Language

The atomic elements of the license recombination and description language are basic licenses provided by the system, defined by developers, or third party licenses. These licenses are internally described by XML documents for storage, transmission and handling purposes. Licenses provided by developers must implement a handling interface that enables the gatekeeper to use this license. If the license uses other components, it is inevitable that interfaces for these components are provided. Those licenses must be described by XML documents. We use a model driven architecture approach to model the licenses and derive an XML document from that model to store, transmit and handle licenses. The third party licenses are represented by Grid services or plug-ins that provide an interface to gather information about licenses (i.e. free/used licenses) and obtain one or more licenses. This interface must be able to derive an XML document describing the license. The basic licenses provided by the system are: node/organization/network locked, date/time-based (such as the service can be used until day x, the service can be used x times or the service can be used x minutes/hours/-days), user-based and based on special parameters like CPU cycles, memory-usage and data transfered. The license recombination and description language allows to recombine basic licenses to form complex licenses and to form recombined complex licenses. There are three main constructs for recombination: negation, fork and chain. Each complex license contains one or more negations, forks, chains, recombined or basic licenses. The chain allows to define a sequence of conditions that must be fulfilled. The fork operation can be used if only one of two or more conditions must be satisfied. The negation operation allows to negate any other license element. All these licenses can be assigned to the container, to services and more fine granular to methods, and in case of Grid services to service resources.

4 Implementation Issues

Currently, we have a prototype implementation of the license manager and the license recombination and description language. In listing 1, an example basic license is shown. Listing 2 shows a more complex license which is a recombination of basic licenses using fork and chain rules. The language is still under development and may change over the time.

Both gatekeepers are Axis handlers plugged into the request handling chain. This handlers intercept SOAP messages. On the client-side, this handler extends the messages with additional information to identify the caller, the called service and additional parameters. The handler at the service-side extracts this information and hands it over to the license manager. This component requests the licenses linked to the called service from the service registry and loads them from the license models component. The license manager interprets the license and checks if access is allowed. If so, it invokes the used license component and the monitoring unit and hands the necessary information over. The monitoring unit gets also invoked if access is not allowed.

```
<License>
  <basic name="Date-Range"
         id="1"
         start="2007-01-01"
         end="2007-31-12"/>
</License>
```

Listing 1: Basic license example: Date Range based license

```
<License>
  <chain>
    <fork>
      <basic name="IP-Range"
             id="2"
             start="192.168.0.1"
             end="192.168.0.10"
             subnetmask=""/>
      <basic name="IP-Range"
             id="3"
             start="192.168.1.1"
             end=""
             subnetmask="255.255.255.0"/>
    </fork>
    </basic ref="1"/>
  </chain>
</License>
```

Listing 2: License example: IP Range based recombined license

To integrate a third party license system into our license management system, we have developed a Grid service that periodically gathers information from a local flexLM

server. This information is stored in an internal data structure and registered to the local Globus toolkit 4 monitoring and discovery system. Registering this information to the MDS has a great advantage: access to the information is standardized, and by registering the local MDS to a another "higher-level" MDS (MDS in a grid are normally hierarchically ordered as a tree) the information is spread over the Grid and Grid-wide accessable at one point. A Grid-wide working (meta-)scheduler or other Grid components could use this information for scheduling and other concerns. This periodically updated information includes available and used third party licenses. An example is given in listing 3. This example is an extraction of the output of the following wsrf-query:
`$GLOBUS_LOCATION/bin/wsrf-query -s http://localhost:8080/wsrf/services/DefaultIndexService "/*"`

```
[...]
<ns11:AggregatorData>
        <ns1:Licenses xmlns:ns1="http://grid.fb05.de/wsrf/
            flexLM/FlexLMService">
[...]
  <item xmlns="">
    <expires>31-dec-0</expires>
    <inUse>2</inUse>
    <issued>2</issued>
    <name>pgcpp-linux86-64</name>
    <server>rubens.hrz.uni-siegen.de</server>
    <users>
      <item>
        <address>rubens.hrz.uni-siegen.de</address>
        <licenseType>floating license</licenseType>
        <startTime> Thu 1/19 11:03</startTime>
        <username>doernemk</username>
        <version>v5.1</version>
      </item>
      [...]
    </users>
    <vendor>pgroupdA</vendor>
    <version>5.100</version>
  </item>
[...]
```

Listing 3: Extract from a mds wsrf-query showing one of the registered flexLM licenses

Our licensing system uses the gathered information to detect whether a third party license is available because this license could be part of one of our licenses or necessary for a service or other software. We are currently implementing methods to reserve a flexLM license through this service to use it as basic license in our system. Furthermore, at this moment we have not implemented the end-user clients, the accounting and billing connector and the scheduler.

5 Related Work

As already mentioned D'Andrea et al. have pointed out the importance of developing appropriate licensing schemes for web services [1]. The paper explicates the basic motivation of licensing web services and characteristics that distinct services from software. This work is only an analysis of the problem and general suggestions to solve it. We agree with the authors with respect to the importance of licensing web services and think that licensing is even more important for Grid services to protect the interests of service providers and developers. In contrast to this more conceptual work, we are providing a prototypical implementation.

Jena et al. [2] focus on identifying a collection of services that are aimed at automating the digital licensing of agent-based services. This digital licensing and the agents are implemented as web services. This work also points out the importance of licensing web services. The authors suggest a digital rights management service that uses Open Digital Rights Language (ODRL) [3] to express the digital rights for the agent system. ODRL provides one variant of the Rights Expression Language (REL) which is machine-readable and uses terms to express rights and permissions from a Rights Data Dictionary. This REL can be expressed in XML. The specification of ODLR describes itself as follows: "The Open Digital Rights Language (ODRL) is a proposed language for the Digital Rights Management (DRM) community for the standardization of expressing rights information over content. The ODRL is intended to provide flexible and interoperable mechanisms to support transparent and innovative use of digital resources in publishing, distributing and consuming of electronic publications, digital images, audio and movies, learning objects, computer software and other creations in digital form.". The intention of our language is quite similar to ODRL but we focus on the demands of Grid services. This focus make it easier to read, understand and handle documents. ODRL is intended to cover a wide range of content and because of that it is a really complex language.

Xiaoshe et al. [4] have designed and implemented a floating license related software sharing system (Floating License Sharing System) for Grid environments. The main goal of this solution is the dynamic Grid-wide binding of existing proprietary software licenses (like FlexLM, iFOR/LS and LSF License Scheduler) and hardware at runtime. In general, floating license protected software authorizes its execution by connection to a license server in the local area network. The Floating License Sharing Systems allow to authorize across the Internet. In contrast to our solution, Xiaoshe et al. deal only with floating license protected software, those licenses are normally used to protect the interests of software developers. The protection code is added in the development process and the license is needed to execute the compiled program. The Floating License Sharing System is a fine solution to execute proprietary binary code everywhere in the Grid, but not to protect the interests of software providers. Our solution focuses on the execution of or access to Grid services and the interests of software providers and developers. Our licenses are configured on demand by the administrator using our License Recombination and Description Language. The access conditions specified by the license can vary from user to user.

Guofu et al. [5] propose an approach for organizing and managing floating software licenses combined with a software sharing system in Grid environments. Their license manager is independent of specific local license management systems. This work combines a reservation mechanism with a backfilling algorithm to enable their software sharing system and license manager to get licenses efficiently. This paper is based upon the work of Xiaoshe et al. [4] and uses the Floating License Sharing System.

6 Conclusions

We have presented an approach to license the use of Grid services, with a prototypical implementation for the Globus toolkit 4. Our solution includes a language for describing and recombining licenses. The architecture of the licensing system and implementation issues were presented.

There are several areas for future work. Currently, our solution is a prototypical implementation that should be completed and extended by additional functionality. In particular, the connection to Grid authentication and authorization systems is one of the planned next steps. Furthermore, a detailed evaluation of the Open Digital Rights Language (ODRL) should be made to decide whether we can integrate it into our language as a new basic license type or whether ORDL should be extended to handle Grid licensing issues. Since we are currently focusing on a Grid service container but usually there is more than one container in a Grid, we plan to provide a Grid-wide service usage licensing system that enables to manage licenses for all Grid nodes of a virtual organization. This solution will be extended by a market place where users and service providers can trade licenses. Finally, the relationship between licensing the use of Grid service and the field of service level agreement should be investigated.

Acknowledgements

This work is financially supported by the German Federal Ministry of Education and Research (BMBF) (D-Grid Initiative, InGrid Project).

References

1. Vincenzo D'Andrea and G. R. Gangadharan. Licensing web services: The rising. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, page 142, Washington, DC, USA, 2006. IEEE Computer Society.
2. Rishikesh Jena and Craig Thompson. Digital licensing service for agents and web services. In *IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS-05)*, pages 418–421, 2005.
3. Open Digital Rights Language Initiative. http://odrl.net.
4. Dong Xiaoshe, Wang Yinfeng, Zhengfang Guohua, Yang Shuncheng, and Wu Weiguo. Floating license sharing system in grid environment. *Proceedings of First International Conference on Semantics, Knowledge and Grid*, 0:96, 2005.
5. Feng Guofu, Wang Yinfeng, Guo Hua, and Dong Xiaoshe. Research on software license manager and sharing system in grid. *In Proceedings of Grid and Cooperative Computing Workshop*, 0:35–38, 2006.