Noname manuscript No.
(will be inserted by the editor)

License Management in Grid and High Performance Computing

Yona Raekow · Christian Simmendinger · Ottmar Krämer-Fuhrmann

Received: date / Accepted: date

Abstract Commercial codes usually come with a licensing mechanism in order to restrict their use. The lack of license management schemes is becoming a major obstacle for the commercial exploitation of existing Grid infrastructures. In this paper we present a complete license management architecture that enables a pay per use license management, which can be deployed together with an on-demand computing scenario. In order to cover the entire license management process, a complete Grid-friendly license management architecture was implemented and is presented here. The license management architecture we present is based on the requirements we collected from 18 real buisness applications [1]. The Grid-friendly solution that we developed has various benefits that we believe will lead to a fast adoption by independent software vendors (ISVs) and their clients and hence pave the way for commercial applications in grid environments.

The authors are partially funded by The BEinGRID project. The BEinGRID project has been funded with support of the European Commission in the Sixth Framework Programme (FP6) as part of the Information Society Technologies (IST) priority.

Yona Raekow
Fraunhofer SCAI
Schloss Birlinghoven, 53754 Sankt Augustin
Tel.: +49-2241-142927
E-mail: yona.raekow@scai.fraunhofer.de

Christian Simmendinger
T-Systems Solutions for Research GmbH
Pfaffenwaldring 38-40, 70569 Stuttgart
Tel.:+49-711-6862333
E-mail: christian.simmendinger@t-systems.com

Ottmar Krämer-Fuhrmann
Fraunhofer SCAI
Schloss Birlinghoven, 53754 Sankt Augustin
Tel.: +49-2241-142202
E-mail: ottmar.kraemer-fuhrmann@scai.fraunhofer.de

Keywords Grid Computing · License Management · High Performance Computing · FLEXNet

1 Introduction

Over the last few years, Grid technology has evolved from a technology designed largely for the research and open source community towards an open framework supporting the general business domain. This development posed various new challenges to Grid computing. In order to tackle these problems and foster the adoption of Grid technology in European business and society BEinGRID [1] has gathered the requirements for a commercial Grid environment. One of the key elements is a pay-per-use license management scheme for Grids. License management has until recently not played a role in Grid computing. But in order to use Grid in a commercial setting it is unavoidable to introduce a flexible pay per use license scheme. To our knowledge, the solution we present here is the first complete license management architecture for Grids.

1.1 The current license management situation

Especially small and medium enterprises (SME) from the engineering community stand to profit from pay-per-use HPC Grid scenarios. Very few of these SMEs however maintain their own simulation codes. Commonly commercial codes from independent software vendors (ISV) are used with associated licensing. The licenses for these codes are typically issued on a yearly basis: Customers buy a fixed number of licenses, with associated features and included support. The generated revenue for the ISV therefore is predictable and stable. In addition this business model guarantees that

the provided codes are always in line with the requirements of the end-users: There is a close dialog between ISV and end-user.
Unfortunately, the business model is in contrast to a pay-per-use scenario. In a pay-per-use scenario there is no predictable revenue for the ISV and unless the ISV is also the license service provider (LSP) the ISV would loose the direct contact to their end-users. The currently established business model implies a substantial over-provisioning of licenses: End-users need to buy more licenses than they require on a daily basis in order to satisfy their peak requirements. With a pay-per-use scenario this over-provisioning immediately becomes obsolete with a corresponding loss of revenue for the ISVs. Therefore nearly all attempts to convince ISVs to agree to provide pay per use licensing schemes that can be used in a Grid environment have been unsuccessful in the past.
On the other hand a pay-per-use model would create a new source of revenue for ISVs, because SMEs, which so far could not afford to purchase licenses can now access the licenses on a pay-per-use basis. Additionally large customers become able to dramatically increase the number of licenses during peak-demand periods. These contrasting business models make a direct transition towards a pay-per-use license model on a new technology basis rather difficult, even more if it requires a change of technology with respect to license management. In oder to convince the ISVs of a new business model it is necessary to provide them with a non-interruptive transition on the basis of currently used technology. Almost all of the relevant HPC simulation codes from ISVs (and these are the potential candidates for a Grid scenario) rely on client-server based license models and the overwhelming part of them make use of the FLEXNet licensing, available from Acresso [2] (formerly Macrovision). This strong market position and customer acceptance of FLEXNet made it imperative to support its license models in a Grid environment.

1.2 Related Work

To the best of our knowledge there are currently two other projects that are developing a license management architecture for Grid environments ([9], [8]). Both architectures do not rely on a client server based license mechanism but rather develop a technology that allows to be independent from a license server and therefore does not require a connection to a license server during runtime. While these approaches allow more flexibility, they also require that the ISVs modify their code in order to incorporate the license mangement into their software. A short comparison between our approach and the approach in [8], is discussed in [**?**].

1.3 Organisation of the Paper

In section 2 we will provide an overview of the architecture. Section 3 will give an overview of the tools used and go more into implementation details. Finally in section 4 we present some concluding remarks.

2 Architecture

In this section we present the requirements that have been elicited from 18 business experiments in the BEinGRID project with respect to license mangement. From these requirements we identified capabilities which led to a license management architecture that we will introduce in this section.

2.1 Requirements of a License Management Architecture

The following license related requirements have been elicited:

Gridification of Currently Used License Management Systems: Supporting a pay-per use license management implies support for all major Grid middlewares: GRIA, GT4, UNICORE and gLite. The commercial codes for which requirements were identified were all based on the license mangement for FLEXNet. To avoid supporting many different middlewares we designed a generic middleware independent solution, which would support an arbitrary client-server based license scheme including FLEXNet. In fact the solution is even usable in a non-Grid context and hence is able to cover various different scenarios.
There are some side-implications with respect to accounting and billing associated with the above requirement: Whereas in the non-Grid scenario the bill already has been paid for in advance and therefore accounting plays a minor role, the pay-per-use model needs to support a flexible cost unit based accounting rather than an identity bound accounting: The reason is that usually institutions or research groups own the licenses, not their individual members.

Limited LSP Capability: The required fluent transition to a pay-per-use model with respect to the underlying business model made it likely that at least for a short transitional period (where short can mean: up to a decade) the ISV will also assume the role of the license service provider (LSP) for all its customers: ISVs need to be able to quickly implement and refine the evolving new business models. This immediately implies that scalability of the license service would become an issue. In this transitional period the ISV would have to maintain potentially several thousand simultaneous connections to the FLEXNet server. (FLEXNet itself is able to handle this amount of connections).

Grid License Model: The FLEXNet scheme (floating licenses from a single possibly redundant server) or other client-server schemes are limited in their scope and scalability. An extension of such a scheme to a scenario where licenses are offered as standard resources in the Grid therefore seems to be a logical step. This scenario readily implies that licenses have to be scheduled like other resources e.g. networks or CPUs. Licenses are typically an even more precious and expensive resource than compute-power. It therefore is exremly important that licenses are not unnecessarily blocked by queued jobs or that jobs, which already have allocated their computing-resources do not get blocked by missing license-keys. Interfaces to e.g. Grid schedulers therefore are needed. An extension of that scenario, where license owners are allowed to re-sell their licenses on a LSP basis should be feasible with respect to technology. This scenario extension would, however, certainly require the cooperation of ISVs from a business perspective. An even bigger step for the ISVs but maybe a logical one would be to entirely drop the concept of selling a yearly license and to instead introduce a – probably token based system, where licenses are accounted and billed on a per job basis.

2.2 Capabilities

Extension of Job Description and Submission: This capability covers the extension of the job description and its submission with respect to license management. A user needs to provide details about the requested licenses, including authorization as well as the accounting context. This capability allows a user to request license resources in a similar manner as currently implemented in Grid middlewares for any other resource (CPU, memory, etc). The resources here can be either own licenses, licenses provided by the service provider or an external LSP.

Authorization for Job Submission: This capability covers authorization mechanisms required with respect to license resource requests. The sole deviation from current standard authorization mechanisms is the type of resource, namely whether or not a user is entitled to use a specific license server, specific features of the licensed software or whether limits exist with respect to the number of licenses. In complete analogy possible solutions range from simple locally maintained lists of PIN/ TAN mechanisms up to a full integration into identity management systems like Shibboleth or VOMS with explicit requests to home organizations of the users and/or third party license service providers or even requests to license Brokers.

Resource Management Extension: This capability covers the extension of a local resource management system. In a batch prologue the resource management can dynamically reconfigure the license proxy in order to grant access for a specific userID at the assigned ideally dedicated local resources. Correspondingly in a batch epilogue the proxy is reconfigured in order to prohibit non-authorized access. At this point security of access to the license server is transferred from a e.g. certificate based authentication/authorization level to the required network level security at which e.g. FLEXNet operates. However, this is only required if there is no additional run- time authorization at the license server.

License Proxy: All external accesses from dedicated local resources are re-routed via the proxy, which can allow or reject these connections (see Resource Management Extension) and then possibly re-routes this request via a proxy-chain (including run-time authorization) to the remote license server. The proxy and possibly its upstream counterpart both log access time, duration and accounting context.

Accounting and Billing: This capability covers the accounting and billing of licenses. In order to produce the complete accounting log, information from both the proxy (userID / time-stamp) and the license server (userID / time-stamp, number of licenses, license features) are required. The actual details of billing and accounting are not only depending on middleware, but also on the underlying business model. Depending on whether licenses are owned by the user, the service provider or an external static LSP is obtained via a Grid broker, the exchange and assembly of the actual accounting and billing information will differ.

Encapsulation of License Server: This capability relates to the integration of existing license servers. It not just addresses a possible encapsulation as a web service but more generally an integration of the license server into the respective Grid middlewares. Remark: In our license management architecture implementation the upstream proxy partly provides this encapsulation. (Run-time authorization)

2.3 Architecture

The license management architecture components (that are: LM-Job Description and Submission, LM-Authorization, LM-Proxy, LM-Monitor, LM-Accounting) are designed to provide a complete license management for FLEXNet (or more generally: client-server license models) based codes by Independent Software Vendor (ISV) in Grid environments. Job submission and description are extended with respect to the list of resources (license server) and possibly required authorization credentials. License authorization is then performed at submit time via a query to a remote service. In a first step license authorization was based on a local access list (ACL). Since this latter functionality is frequently required for local usage in a non-Grid context, (e.g. any linux cluster which is shared between different organizations) we will maintain two different versions of license authorization. In a second step an optional/alternative run-time authorization (at the remote upstream proxy) via a PIN/TAN mechanism is implemented. For details of this implementation we refer to section 3. Usage of licenses is accounted by the LM accounting module. The LM monitor monitors the status of available licenses and upon request returns this status to higher level services like license schedulers/brokers or an SLA monitor.
Figure 1 shows an overview of how the components of the license management architecture are related to each other. A job is submitted via some job submission and description mechanism, this can be the client of a grid middleware or a portal or simply via locally from the compute resource. Once the job is submitted it is retrieved by the resource management of the compute resource. Currently under development is a monitoring system that checks, before a job is entered into the queue, whether the number of licenses that are requested by this job are available. Once the job starts running and gets to the point where a license is requested, it connects to the local proxy server that makes sure that a user is allowed to access the license server. The request is then routed via a proxy chain to the the upstream proxy that connects to the license server specified by the job. The license is issued and some logging information is written, which can be used by the accounting component. Of interest is, which license has been requested, which feature has been requested and for how long the license has been used. Another component is the LM Authorisation web service component that handles the license management related credentials of the user of the system.

3 Implementation

3.1 Overview

Industrial environments typically rely on commercial applications of ISVs with an associated License Management - very common is FLEXNet from Acresso [2], which is the de-facto standard in this area. FLEXNet has a closed API, is proprietary and based on a simple client-server mechanism. The FLEXNet scheme allows floating licenses, which are not bound to a specific host. Rather they are allocated dynamically to arbitrary hosts. Licenses then are checked out at the license server when an application starts and checked in when it ends.
In principle FLEXNet therefore is suitable for usage in a Grid environment. There are, however major security and identity issues with respect to the access to the license server in a Grid environment. For example the FLEXNet software is able to filter legal and illegal accesses based on the host IP, but is not able to grant access on the basis of user/group certificates. This implies that on every Grid site an unauthorized user could check out and use an arbitrary number of licenses once the corresponding license server is exposed.
In order to support this standard (or in general: any client-server license management scheme) we propose to transparently reroute the encrypted socket-based communication between client and server via a SOCKS proxy-chain. The communication from the license client then can be transparently forwarded (via a socksified job shell) to a remote upstream proxy and then to the remote license server. The run-time authorization at the upstream proxy is handled via a PIN and associated encrypted one-time passwords (TANs). The PIN here represents a license account and can be used to provide the accounting context. License owners (typically institutions) can set up an arbitrary number of these license accounts under a billing account. This mechanism allows institutions or research groups to share access to licenses and to use licenses in a cost unit based accounting context. A self-imposed budget-control for pay-per-use scenarios will be available. The handling of the one-time passwords (generation of TAN lists, l icense accounts and their properties) is implemented as a webservice. A design of a portal which enables users to access these web services, conveniently share accounts, automatically extract one-time passwords and submit correspondingly modified jobs is currently in progress.

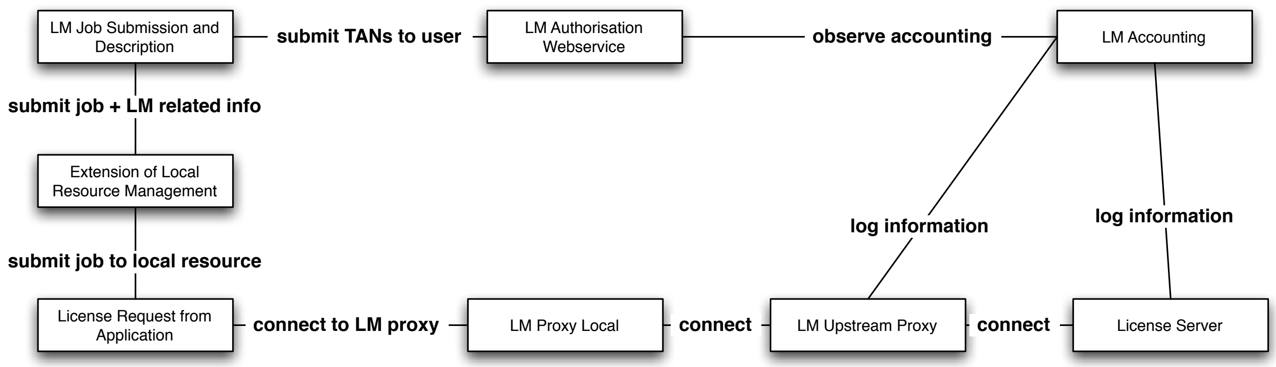

LM Job Submission and Description
submit TANs to user
LM Authorisation Webservice
observe accounting
LM Accounting
submit job + LM related info
Extension of Local Resource Management
log information
log information
submit job to local resource
License Request from Application
connect to LM proxy
LM Proxy Local
connect
LM Upstream Proxy
connect
License Server

Fig. 1 Overview of the license management architecture.

The local SOCKS proxy (at the service provider site) additionally can be re-configured on a per-job basis by the local resource management systems, thus providing an additional layer of security. The license management architecture is complemented with extended functionalities for job submission and description, license accounting, billing and a License Monitor. Information from FLEXNet itself here needs to be synchronized with information from the authorization module in order to provide a complete cost unit based accounting.

To sum up we note that the entire system consists of four entities: The provider of compute resources, a SME, an engineer at the SME and a license provider. The SME can have various license accounts that correspond to internal projects, this facilitates their internal accounting and monitoring of expenses. Hence, these license accounts are not managed directly by the engineer at the SME, instead the engineer is provided with the account and a certain number of one time passwords that he has to his disposal for submitting jobs that require a license to the compute resource provider. Once the job runs at the provider site and the license is required a connection via the proxy chain to the license provider is established.

3.2 Proxy

The core element of LM-Proxy is a standard SOCKS 5 proxy (Circuit Level Gateway), which is dynamically reconfigured by the resource management system on a per-job basis. The proxy then only permits access from authorized_users@allocated_nodes. A first prototype was developed that is based on local ACLs (Access Control List - is a list of permissions attached to an object) and targets local usage e.g. Linux clusters which are shared between organizations. Provided an out-of-bounds trust relationship between license server and service exists, it also is suitable for usage in a Grid context.

The user provides a license account name with corresponding one time passwords which are validated before the proxy can be accessed.

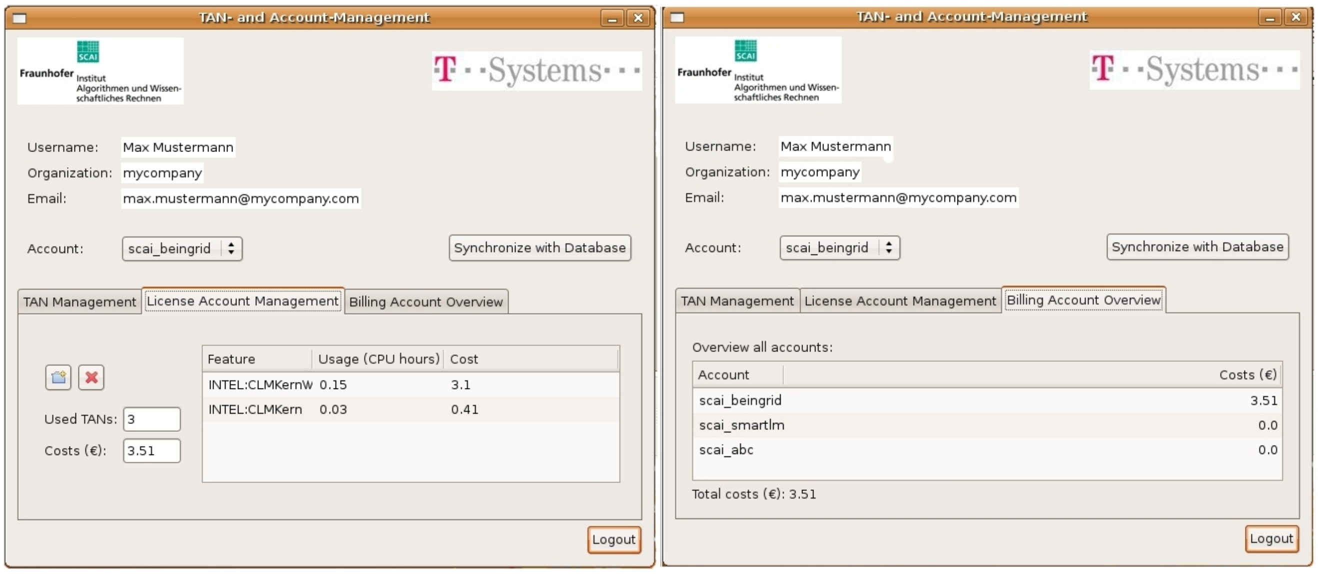
For our purposes we identified the tsocks [4] implementation to be the most suitable, since it allows a chaining of proxies.

The main advantage in using a socks based approach, rather than encapsulating the TCP packages of the FLEXNet server as SOAP-messages, is scalability.The socks server is multithreaded and if required allows preforked processes. According to the project webpage [11] the socks server allows up to 2500 connections per second.

3.3 Webservices

Part of the LM implementation are two webservices. One is is responsible for authorisation and handles the one time passwords. The second one is responsible for accounting, i.e. it provides functionalities, s.t. the user has a cost overview. The web services are created using Axis2 [6] and deployed at the site of the resource provider. The corresponding clients run on the site of the SME Fig. 2. Currently the client consists of a graphical user interface that allows convenient management for all license accounts that a SME holds for a given system provider site. Under development is the embedding of the client into a GridSphere [7] portal, which is used by many system providers to provide access to their resources.

TAN Web Service: This web service handles the one time passwords that are submitted together with the job description. It provides functionalities to create new TAN lists, to block and unblock TAN lists and to check how many TANs have been used from the current list.


TAN- and Account-Management
Synchronize with Database
TAN Management
License Account Management
Billing Account Overview
Logout

Username: Max Mustermann
Organization: mycompany
Email: max.mustermann@mycompany.com
Account: scai_beingrid

Feature	Usage (CPU hours)	Cost
INTEL:CLMKernW	0.15	3.1
INTEL:CLMKern	0.03	0.41

Used TANs: 3
Costs (€): 3.51

Overview all accounts:

Account	Costs (€)
scai_beingrid	3.51
scai_smartlm	0.0
scai_abc	0.0

Total costs (€): 3.51

Fig. 2 Overview of the graphical user interface.

The core element here is the tool pam_sotp [5] which is a UNIX tool that allows the creation of TAN lists. This tool is encapsulated by the web service. Features of the tool that are accessible via the webservice are the following:

– Create TAN list: This function creates a TAN list, i.e. a list of one time passwords. The user can specify how many TANs he would like to have in his list. The number of TANs corresponds to the number of licenses the user wants to request later on. The pam_sotp tool allows to specify over which alphabet the TANs should be created and which lenghts they should have. These parameters are hardcoded in the web service and are not the choice of the user.
– Block TAN list: If the user realizes that one license account used already to many licenses, he can block the TAN list for this given license account. The TAN list remains unchanged, but all attempts to access is result in an error.
– The user can unblock a previously blocked TAN list. This allows the license account that corresponds to this TAN list to access this TAN list again and it is again possible for this license account to submit jobs that require a license.

Accounting Web service: This web service allows the user to check the license usage in detail for each license account he owns, e.g. which features a license account has used. He also has to possiblity to get a summary of of all the license accounts that he manages and the corresponding license usage costs. The functionalities that the accounting web service provides are as follows:

– Get license accounts, this functionality retrieves all license accounts that a user manages from the server.
– Create license account: allows a user to create a new license account.
– Delete license account: deletes an existing license account.
– Get account info of a selected license account. This information consists of a detailed listing of features for which the user has requested licenses and provides information on how often and how long these licenses were used. It also provides information on how much money a license account already spent for licenses overall. Furthermore a user can see how many TANs have been used and how many are still available.

3.4 Job Submission

The job submission needs to be extended: In addition to the job description the user has to provide his license account name and a sufficient number of one time passwords for the licenses he wants to use. These two additional parameters are stored in the environment variables SOTP_ACCOUNT and SOTP_PASSWORD. Instead of just submitting one one-time password the user can also submit a subset of his TAN list as a file and can therefore access a large amount of licenses conveniently.

4 Conclusion and Future Work

The strong market position and customer acceptance of FLEXNet made it imperative to support the general class of a client-server based license management scheme. We also think that a non-interruptive transition to a pay-per-use model is more realistic than its interruptive counterparts, that are currently proposed in other projects, e.g. [8].

At the same time these client-server based license management schemes are not suitable for use in a Grid, since the security and integrity of the license server immediately becomes compromised in a Grid-context. As an additional problem the protocols of client-server interaction are proprietary with a closed API. In order to provide a secure and reliable solution which overcomes these deficiencies and is still compatible with these license management schemes, we have designed a new license management architecture. The architecture is centered around a dynamically re-configurable SOCKS proxy infrastructure. It is designed to provide excellent scalability. The solution is generic and with minor modifications (job description and submission) compatible with all current middlewares. Additionally it is suitable for LSP in a non-Grid context. Since it also provides a very high level of security we think that this solution will play a very significant part in paving the way towards a pay-per-use license model.

Currently work in progress is to embed the job submission and the web services into a GridSphere portal. The monitoring component, that expands the local resource management is also currently developed. A first prototype of this architecture has already been deployed and is currently evaluated. Several providers of compute resources have already expressed interest in this solution.

A remark on scalability: As mentioned earlier the socks server permits several hundred requests per second cf. [11]. The main reasons why we chose to use socks rather than a SOAP encapsulation of the license-server/ license-client communication was that it has a good performance and is scalable. So the number of licenses that can be checked out is limited by the bandwidth between the license server and the socks server, as well as the performance of the license server itself. According to the FLEXNET end user guide [3] a flex server can handle several hundred requests per second if running on an appropriate machine and is configured properly. In case a single license server is not sufficient, one can run several license servers in the network in order to allow more requests.

References

1. BEinGRID - Business Experiments in Grid, www.beingrid.eu (as of 01.02.2009)
2. Acresso Software, FLEXNet, www.acresso.com (as of 01.02.2009)
3. FLEXNet Licensing End User Guide
4. TSOCKS - Transparent SOCKS Proxying Library, http://tsocks.sourceforge.net/ (as of 01.02.2009)
5. pam_sotp - A module for PAM that provides support for One Time Passwords (OTP) authentication, http://www.cavecanen.org/cs/projects/pam_sotp/ (as of 01.02.2009)
6. Axis2 - http://ws.apache.org/axis2/ (as of 01.02.2009)
7. GridSphere - Portal framework that provides an open-source portlet based Web portal, http://www.gridsphere.org/ (as of 01.02.2009)
8. SmartLM - Grid-friendly software licensing for location independent application execution, http://www.smartlm.eu/(as of 01.02.2009)
9. GenLM - A license management architecture developed in the D-Grid project Partnergrid, http://www.epg.fraunhofer.de/solutions/software/genlm.jsp (as of 01.04.2009)
10. Jiadao Li, Wolfgang Ziegler, Oliver Wäldrich, Daniel Mallmann, "Towards SLA Based Software License Management in Grid Computing", CoreGRID - Network of Excellence, TechReport, TR-0136, 2008
11. SS5 Official Web Site - http://ss5.sourceforge.net/ (as of 10.04.2009)