Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 615-621

Floating License Management - Automation Using Web Technologies

A. Doloca, O. Ţănculescu

Adrian Doloca
Gr.T. Popa University of Medicine and Pharmacy,
Department of Mathematics and Informatics
Romania, Universităţii 16, 700115 Iaşi
E-mail: ad@umfiasi.ro

Oana Ţănculescu
Gr.T. Popa University of Medicine and Pharmacy,
Department of Fixed Prosthodontics
Romania, Universităţii 16, 700115 Iaşi
E-mail: otanculescu@eprpru.umfiasi.ro

Abstract: This paper examines the use of distributed computing based on web services with application to floating license management. The main goal is to automate the processes pertaining to the management activities and ensure at the same time that the security and flexibility requirements are met. We present the challenges posed by these requirements, propose a design and some implementation aspects using the latest .NET development platform.
Keywords: floating license management, distributed system, web service, multithreading, .NET development.

1 Introduction

Software is a vital component of today's businesses. The fast development of the software industry has produced applications for automating virtually every task, thus saving time and finances. At the same time protecting software from unauthorized usage became a strong necessity. Except the public domain that contains material not owned or controlled by anyone, all other software products are subject to the copyright law which gives the author of an original work exclusive right regarding its distribution, publication and modification. The software license is for that matter a legal instrument governing the usage and redistribution of software applications protected by copyright, but at the same time a technical means which will prevent the application from running when no authorization is available. This allows a better control of the product's distribution and ensures that illegal use will be kept to a minimum allowing the producer to economically benefit from the investment and to prevent revenue loss [1].

Concerning the method of protection, two main categories of software licensing systems are most popular: node-locked and floating licenses. Other license types and a method of optimization are presented in [2].

Node-locked licenses allow an application to be used on only one machine. This is achieved by making the license machine dependant. Therefore the license will include some unique IDs of hardware components like hard drives, CPU ID, network card ID, which will be checked as part of the license verification process.

Floating licenses (roaming licenses) allow multiple users to share access to a software product. Usually the number of users simultaneously running the application is limited, this limit being embedded in the license itself. This provides a higher degree of flexibility since users are not restricted to using the application from a designated workstation. The software can be

installed on any number of client computers but if the limit is reached a new user whishing to start it is put on hold and will be able to start its session as soon as an existing session is finished.

FlexNet and the older version *FlexLM* [3] are the de facto standards in electronic licensing being a partner of major software producing companies. These products support a large variety of license management scenarios focusing at maintaining a high degree of flexibility in choosing feature and pricing configurations. Using FlexNet software producers can generate, transfer and activate licenses and can also protect their products against piracy. [4] and [5] deliver an insight into the challenges that floating license management poses and shows how FlexLM finds solutions to these challenges.

This paper presents an original software system that provides flexibility in the management of floating licenses in enterprise environments, attaining a high degree of automation by using the latest Internet development technologies. It also discusses details of implementation and benefits of this system. The presentation is structured as follows. Chapter 2 looks into the features that an advanced floating license management system is required to have. Chapter 3 shows the structural aspects and communication between system components. Security issues generated by exchanging information over public networks are addressed in chapter 4. Some implementation aspects and recommendations can be found in chapter 5. Final conclusions are presented in chapter 6.

2 System Features and Functions

The floating license management system should have at least some features to ensure that the terms of the license contract are enforced and that this is done in a secure manner, with minimum effort by automating the necessary tasks. Below we present these features in detail.

A. Limit the number of simultaneously running application instances according to the license contract

At startup the application will connect to the license server in order to check if it has the authorization to run. If the maximum number of running instances has already been reached, the requesting application is denied the right to start and will close. When an ongoing working session is finished, the corresponding seat on the license server is freed and another application can start. Of course, some requirements exist for this feature:

- the license information must be centralized on a license server;
- the license server must be accessible in a network by all client computers which have the application installed.

B. Ensure security of the license content and prevent fraud attempts

The license content resides on the license server usually in the form of a text or a binary file which contains information pertaining to the company that acquired the license and the maximum number of simultaneously running application instances. In the proposed system the following set of parameters are used:

a) *License-ID*: uniquely identifies the license in the license database residing at the producer of the software. It contains a random number and possibly the creation date and time. This will block, for example, any attempt to use older licenses that have been issued to the same company, for the same software.

b) *Company-ID*: short identifier of the company that purchased this license. It prevents license usage across companies by simply copying the license content.

c) *Application-ID*: identifies the application the license refers to. Different software can use the same license management system by using different Application-IDs. It prevents illegal use of licenses issued for other software.

d) *Configuration-ID*: this piece of information specifies the configuration/version of the software which will be running (demo version, full version, light version, professional version, enterprise version, etc.). It ensures that only the authorized version of the software will be used.

e) *Hardware-ID*: uses one or a combination of multiple serial numbers of certain hardware pieces to ensure that the license is used only on a certain machine that was authorized for license management purposes. The most popular hardware related serials are: the hard drive ID (volume serial number), network card physical address (MAC), CPU ID. This prevents installing the license on several machines and using them simultaneously for increasing the number of running software instances over the authorized limit.

f) *MNRI (Maximum number of simultaneously running application instances)*: specifies how many instances of the software can run at the same time on client computers that use the license server. If this limit is reached, additional instances are prevented from running until an existing session is terminated.

The above listed information is combined into a unique string – the license content, which is encrypted using a strong encryption algorithm and stored in the license file. The encryption protects the license from any attempts to adjust the contained parameters for using it in other environments than the one it was issued for.

C. Provide flexibility for license configuration and installation

For maximum configuration and installation flexibility the license file will be regularly synchronized with a remote server controlled by the software producer. The first time the license server is started it will contact the remote server for the first synchronization. As a result, the license file is created and subsequently used until the next synchronization takes place. During each license synchronization, any modifications in the license parameters on the remote server are automatically transmitted to the license server. In this way no manual updating process of the license is required. This process is depicted in figure 1, in chapter 3. The synchronization ensures that the license used by the client is up-to-date and it hasn't been tampered with. At the same time all contract amendments that are made at one time are quickly and automatically implemented on the client's server.

D. Provide flexibility of working with multiple application types and multiple configurations of a software product

Due to the *Application-ID* and *Configuration-ID* parameters the license is linked to a specific software product and a version/configuration. The same license structure can be used with other software products just by adapting these two parameters accordingly. On the client side, one license server will be able to handle multiple licenses for multiple applications and configurations.

E. Offer flexible administrative functions for the client company as well as for the producer of the software

Regarding the used licenses, the client company is able to perform the following tasks:

- install license during the first synchronization;

- update license information during subsequent synchronizations;
- monitor and manage active users and licenses;
- configure intranet and internet network settings, etc.

The software producer has also an administration tool that enables them to:

- create new licenses according to specifications given in section B of this chapter;
- modify existing licenses (upgrade or downgrade);
- monitor the synchronization state of each client;
- block or activate a license, etc.

3 System Architecture

The architecture of the floating license management system is based on a distributed structure (figure 1) comprising three major components:

1) **Client workstations** running the licensed software;
2) **In-house license server** which authorizes software utilization on client workstations on one hand, and performs license synchronization with the remote server on the other hand;
3) **Remote License Server** controlled by the software producer, hosts the license database and performs automatically synchronizations over the Internet with the clients, acting as a web service.

The distributed structure using an in-house license server and a remote license server ensures centralized administration of the licenses both at the client and at the software manufacturer. Also the license installation and update are done automatically with a minimum of intervention from human operators. This accomplishes two main goals: ease of administration and security of the license content. Other advantages include use of public networks (Internet) and flexibility in case of software upgrade [6].

The client computer sends a message to the in-house license server during startup requesting authorization for software utilization. At the server, a service (e.g. Windows service) receives the message and based on the content of the license residing in the license file and on the current number of running application instances sends back a response message authorizing or denying software utilization. Using a web service client, the license server checks periodically with the remote license server if there are any modifications and, if necessary it updates the license file. If this communication fails, then a warning is issued on the in-house license server but the functionality is not affected. Checking with the remote license server is attempted again and if it fails repeatedly, after a specific amount of time, the license on the in-house license server expires, which results in blocking all software utilization on client workstations. Normal functionality is resumed after a successful communication between the in-house and the remote license servers. This mechanism ensures that the license is up-to-date at all times and that fraud attempts are avoided.

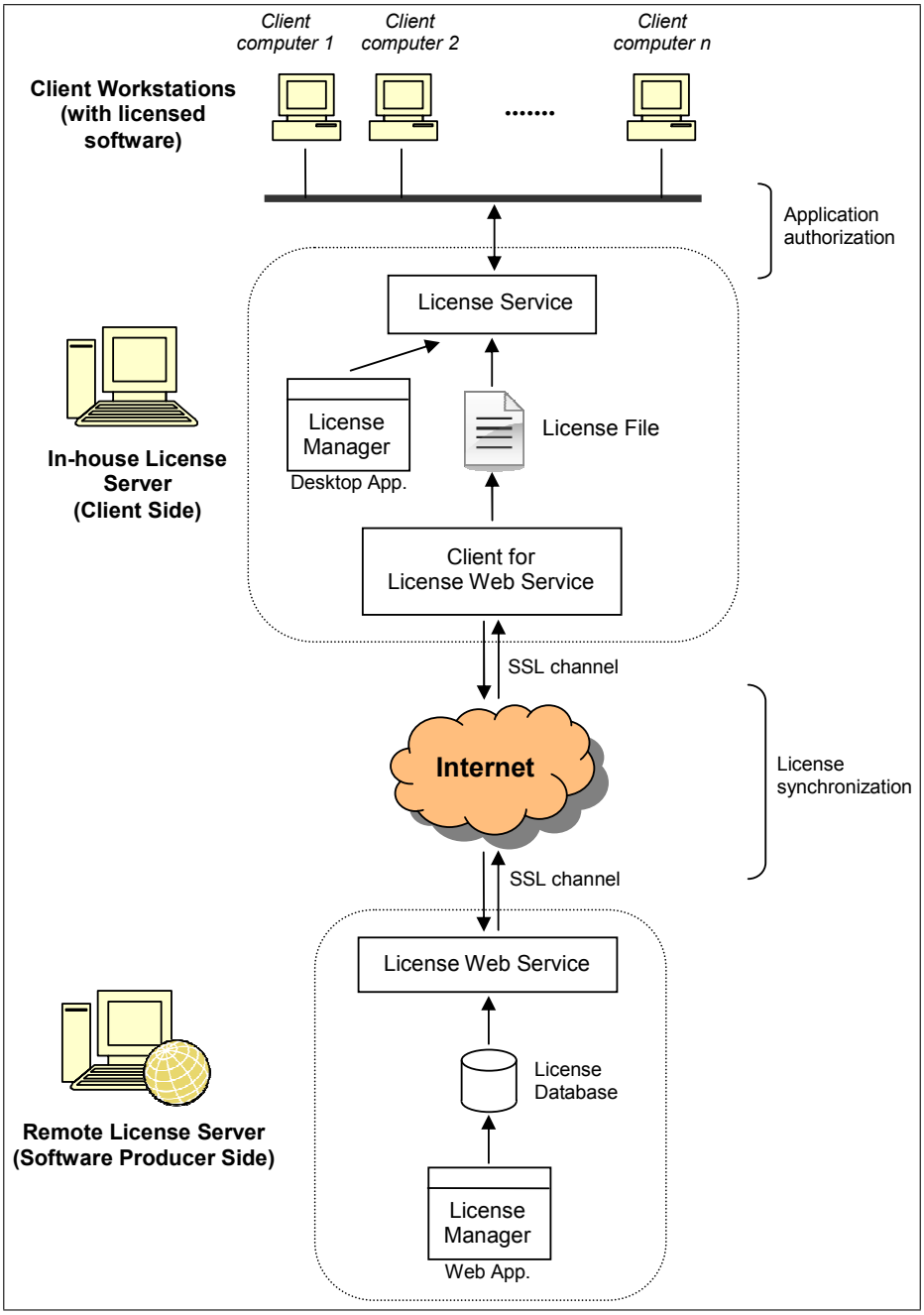

Figure 1: Floating License Management System Architecture

4 Security issues

Being a system that relies on exchanging information over public computer networks and also working with private information (i.e. the license content) in an environment that is not controlled by the software manufacturer, security has an important role. There are two aspects of this problem: securing the data transfer during the synchronization process and protecting the license content.

An effective method for dealing with the first aspect is to create an encrypted communication channel using Secure Socket Layer Connections (SSL). Thus, not only the data sent to the remote license server but also the license content received back, are protected from possible outside attacks. The subject of using SSL for Windows web services is covered in [7].

The information contained by the license file (see chapter 2, section B) must also be protected as it contains sensitive information established by the license contract. Because the information is combined into a unique string, this can be encrypted using an advanced encryption algorithm like Rijndael [8] and stored as a string representation in the license file. The license service will read and decrypt internally this information at startup.

5 Implementation aspects

The overview architecture of the license server is presented in figure 2. As the application has to be able to communicate at all times, it was implemented as a Windows service which works in the background and spawns two additional execution threads. One is the TCP Port Listener thread which receives all the requests from the local network for application start-up authorization and keeps track of the machines and active users that use the licenses. The other thread is responsible for synchronizing the license file content with the remote license server. This structure allows timely responses through both communication channels by keeping the two activities as independent as possible.

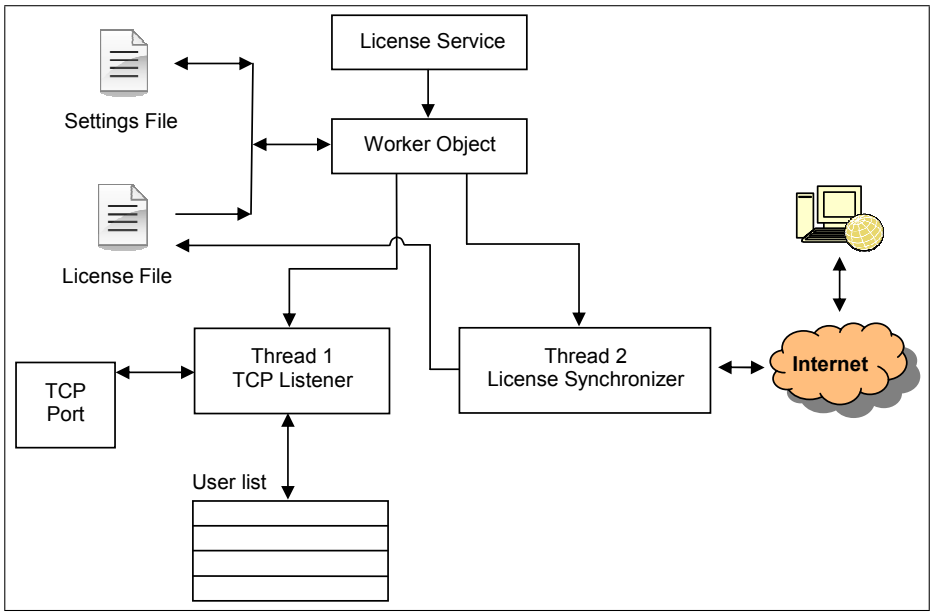

Figure 2: License Server Implementation Structure

We chose for implementation the Microsoft .NET Framework platform using the Visual Studio .NET 2008 development environment. Among the reasons:

- development support for a wide range of technologies like Web service, Windows service, Web applications and desktop applications;
- support for programming of TCP Ports;
- very good documentation.

6 Conclusions

This paper discusses the aspects of floating license management and proposes a specific system architecture that meets the flexibility requirements and offers at the same time, a high degree of automation for most of the license management activities. Using the Web service technology, license installation, activation and synchronization can be performed automatically by the system. Details of implementation using the latest .NET development framework are presented.

Bibliography

[1] R. Addy, Software License Management, *Effective IT Service Management*, pp.263-273, Springer Berlin Heidelberg, 2007.

[2] D. Gull, A. Wehrmann, Optimized Software Licensing – Combining License Types in a License Portfolio, *Gabler Verlag*, Vol.1, N0.4, pp. 277-288, August, 2009.

[3] FlexNet Producer Suite for Software Vendors, http://www.flexerasoftware.com/products/flexnet-producer-suite-software-vendors.htm.

[4] S. Sultan, Floating License Management – A Review of FlexLM, http://wob.iai.uni-bonn.de/Wob/images/36311141.pdf.

[5] FLEXlm End Users Guide, http://web.njit.edu/topics/Prog_Lang_Docs/html/flexlm/all.htm.

[6] H. Wang et al ,Web services: problems and future directions, *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol.1, Issue 3, pp. 309-320, Elsevier, April 2004.

[7] HOW TO: Secure XML Web Services with Secure Socket Layer in Windows 2000, http://support.microsoft.com/kb/307267.

[8] Advanced Encryption Standard, http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.