SOFAnet: Middleware for Software Distribution over Internet

Ladislav Sobr, Petr Tuma
Distributed Systems Research Group, Department of Software Engineering
Faculty of Mathematics and Physics, Charles University
Malostranske nam. 25, 118 00 Prague, Czech Republic
ladislav.sobr@st.mff.cuni.cz, petr.tuma@mff.cuni.cz

Abstract

The paper focuses on software distribution over the Internet, points to weaknesses of the current practice and argues for removing the weaknesses by introducing a ubiquitous distribution framework at the middleware level. The paper describes our design and implementation of such a framework, capable of supporting complex distribution and licensing models.

1. Introduction

The advent of the Internet as a ubiquitous and pervasive communication medium brings forth a diverse spectrum of applications only foreseen a short time ago. One application of the Internet that quickly grows in both popularity and usefulness is the distribution of software. The distribution of software over the Internet exceeds other means of distribution in many important aspects, such as the ease of obtaining and installing the software, or the delay from the decision to obtain the software to the installation. Today, the Internet is used for distributing both complete application packages and incremental application updates. Indeed, software that is indispensable to many systems, such as system security patches and virus database updates, is distributed almost exclusively over the Internet.

The problem of the distribution of software over the Internet is that the practice is fragmented. A typical user of a personal computer running Microsoft Windows will rely on the Windows Update framework to obtain the system security patches, on the Office Update framework to obtain the office application patches, on the Acrobat Updater for updates to the Acrobat Reader, on the QuickTime Updater for updates to the QuickTime Player, on something else for antivirus updates, and on still something else for other applications. This fragmentation hinders a wider use of the distribution of software over the Internet, for every distributor has to invest in designing and implementing a distribution framework. The proprietary frameworks typically use mutually incompatible packaging, distribution and installation mechanisms, which may lead to a wide spectrum of problems, least of which is the duplication of effort in designing and implementing the frameworks. Incompatible packaging can cause a redundant distribution and installation of the same software distributed in different packages. Incompatible installation can cause collisions in overlapping parts of system configuration related to different packages.

We believe that the problems associated with fragmentation of the distribution of software over the Internet are serious enough to warrant creation of a ubiquitous distribution framework. We present our design of such a framework, positioned at the middleware level within the SOFA component system. The design allows us to distribute both complete software packages and incremental software updates in a manner that is seamlessly integrated into both the software and the Internet environment.

We proceed by describing the issues of distribution of software over the Internet in Section 2 of the paper. Section 3 presents our design of the distribution framework within the SOFA component system, for sake of readability starting with a very brief overview of the SOFA component system itself. We close the paper by comparing our design with existing practice in Section 4, and by summarizing our results in Section 5.

2. Internet Software Distribution

The basic premise of the distribution of software over the Internet is that in spite of its invisible and effortless nature, it retains the business nature of a trading deal between a supplier and a customer. This means that a distribution framework must support a complete electronic trading process with procedures such as authentication and authorization of business parties, payment processing, management of licenses, and others. The majority of these procedures, however, should be supported externally, through the use of other frameworks. Only the distribu-

tion of the software and the associated management of licenses need to be supported internally by the distribution framework itself.

With respect to the distribution of software, two different environments must be considered. One is the Internet as a loosely-coupled global network, the other is the environment of the intranets as tightly-coupled local networks connected to the Internet. From the point of view of the Internet, which has no uniform administration, the intranets often appear as monolithic entities that represent a supplier or a customer. From the point of view of the intranets, which typically have a uniform administration, the internal arrangement becomes visible. For the intranet of a supplier, this arrangement describes e.g. how the supplier controls the development and external distribution of software. For the intranet of a customer, this arrangement describes e.g. how the customer controls the internal distribution and installation of software.

On both the Internet and the intranets, two traditional models of distribution need to be supported. One, dubbed the *push model*, represents a scenario where a supplier causes the software to be transported to a customer, while the other, dubbed the *pull model*, represents a scenario where a customer requests the software to be transported from a supplier. Both models have their strengths and weaknesses – the push model is preferable for e.g. an automated delivery of software to hosts under uniform administration, but performs badly when the recipient hosts are not always online, whereas the pull model is suitable for e.g. a delivery of software on demand, but performs badly when the recipient hosts do not know what software to demand.

With respect to the management of licenses, a variety of licensing models must be supported in an effective manner. Several licensing models can be found for example in [9]. One of the important licensing models is the *node-locked licensing*, where a license allows the use of software on a specific host only. Another is the *site licensing*, where a license allows the use of software on any host owned by the customer. Yet another is the *as-a-book licensing*, where a license allows the use of software in a certain number of physical copies, or the *concurrent licensing*, where a license allows the use of software in any number of physical copies provided only a certain number of those copies can execute concurrently at any given time.

3. SOFA Distribution Framework

The many-faceted requirements on the distribution of software over the Internet point to a distribution framework that is situated at the middleware level. This position allows the distribution framework to remain separate from the software that uses it, as well as from the underlying operating system. At the same time, the distribution framework remains easy to integrate into both the software that uses it and the Internet environment.

The distribution framework should be open and extensive, rather than providing an exhaustive interface that would cover all the processes needed for a complete electronic trading process. We believe that an extensible framework is always a better choice than an exhaustive interface, because the requirements on the interface are difficult to predict, and even if that were done, the resulting implementation would be huge.

We implement the distribution framework as a part of the SOFA component system [3][11] and refer to it as SOFAnet. The choice of the SOFA component system does not restrict the principal features of our design, which stays independent of SOFA, and provides support for components, which allows for incorporating additional features such as delivery of components on demand.

The dominant feature of the SOFA component system is the component model. An application in SOFA is a *component*, either directly implemented in the underlying environment such as C++ or Java, or recursively assembled from other components interconnected as described by an *architecture*. The components are identified by unique names and versions.

The runtime environment of the SOFA component system resides on hosts called SOFAnodes. A SOFAnode contains a *repository* of components [7] and can launch component applications by obtaining the components from the repository and instantiating and interconnecting them as described in the architecture of the application. Installing, uninstalling and upgrading of a component application or its part is done simply by populating the repository of the SOFAnode.

For the purpose of distribution, data is packed in *bundles*. A bundle is an arbitrary set of components, typically all the components of an application, or those components of an application that form a patch or a plugin. A bundle has a name and a version and serves as a distribution and licensing unit of SOFAnet.

3.1. Bundle Licensing

Due to the business nature of a trading deal between a supplier and a customer in SOFAnet, the relationship of the two parties is defined by a business agreement, here denoted as a *contract*. The contract is a binding agreement between the supplier and the customer, which originates outside SOFAnet, and whose realization consists of a series of steps including negotiation, signing, payment and distribution.

It is important to note that of all the steps involved in a realization of a contract, SOFAnet is only responsible for distribution. Indeed, SOFAnet is only concerned with the part of the contract that describes the distribution of software and the associated management of licenses, which we call the *distribution policy* and the *licensing policy*. The entire contract may even exist only on paper or in another form that SOFAnet does not understand, as long the distribution policy and the licensing policy can be extracted from the contract for use by the distribution framework. The two policies thus represent the contract in SOFAnet.

To illustrate the working of SOFAnet, we use an evolving example. The example starts with a contract where CUSTOMER purchases a license to install an unlimited number of copies of APPLICATION from SUPPLIER. SUPPLIER agrees to distribute new versions of APPLICATION to CUSTOMER for one year. CUSTOMER agrees to execute at most five copies of APPLICATION concurrently. Let us further presume that SUPPLIER has an intranet with two SOFAnodes, one used for development and one for distribution, and that CUSTOMER has an intranet with many SOFAnodes, one used for distribution and the rest for execution.

Extracting the licensing policy from a contract is a delicate problem because a potentially vague licensing policy representation in the contract must be converted into a necessarily specific licensing policy representation that SOFAnet can implement. We have opted for representing the licensing policy by *license objects* that serve as the least common denominators of a variety of licensing models. A license object consists of an identification of the bundle to be licensed, a time period of the license, a number of copies licensed, and a type distinguishing concurrent and as-a-book licensing of the copies. A single license object can describe simple licensing models such as site licensing, as-a-book licensing or concurrent licensing, several license objects can be combined to describe complex licensing models such as shareware licensing with limited evaluation period.

A license object that licenses several copies of a bundle can be split into several license objects, with the total number of copies licensed by the new license objects equaling the number of copies licensed by the original license object. A license object is split in this manner whenever physical copies of the bundle are created.

In our evolving example, the license object extracted from the contract specifies “Application” as the bundle identification, “unlimited” as the time period, “five” as the number of copies and “concurrent” as the type.

We should point out that although SOFAnet keeps track of the licensing policy, it does not deal with license enforcement. In an environment where the customer has full control of both software and hardware, it is not generally possible to enforce licenses. Attempts to circumvent this principal limitation require at least a part of software and hardware of the customer to be under a full control of the supplier, which does not make sense in an open system such as SOFA. This, however, does not preclude the use of digital rights management and software protection systems in SOFAnet. With respect to license enforcement, SOFAnet is simply no better, but also no worse, than other systems.

3.2. Bundle Distribution on Internet

Extracting the distribution policy from a contract represents the same problem as extracting the licensing policy. A potentially vague distribution policy representation in the contract must be converted into a necessarily specific distribution policy representation that SOFAnet can implement. Apart from the identification of the bundles to be distributed, the distribution policy representation must name the parties in the bundle transport, both for the case where the bundles are pushed by the supplier to the customer and for the case where the bundles are pulled by the customer from the supplier and the time period when the bundles can be transported. We have decided to represent the distribution policy by *distribution rules* that configure the actions of a SOFAnode when transporting bundles.

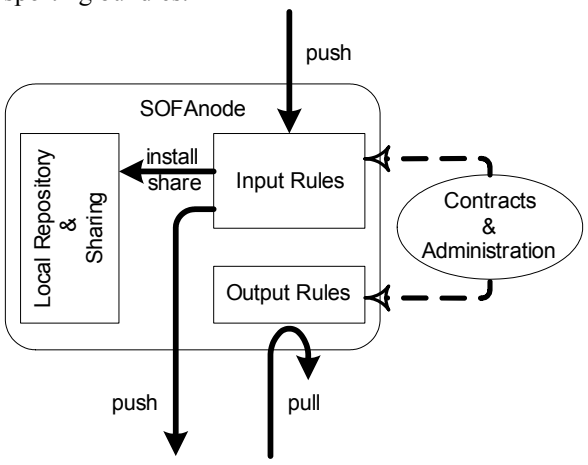

Figure 1. Distribution rules on a SOFAnode

The distribution rules are defined in two distinct sets. The set of *input rules* says what to do when a bundle arrives at a SOFAnode, while the set of *output rules* says what to do when a bundle is requested from a SOFAnode. A distribution rule identifies the bundles to be distributed, the time period when the rule is valid and the action to

perform when the rule is triggered. The actions include principal actions of pushing the bundle to another SOFAnode and allowing another SOFAnode to pull the bundle, as well as auxiliary actions such as sharing or installing the bundle. The position of the distribution rules on a SOFAnode are illustrated on Figure 1.

The distribution rules are primarily used to implement the distribution policy on the Internet, where the input rules implement the pushing of bundles driven by a supplier, and the output rules control the pulling of bundles driven by a customer. The rules can also configure a SOFAnode of a reseller, which forwards bundles from a supplier to a customer. All such rules are derived from the contract that defines the relationship between the supplier and the customer.

In our evolving example, the distribution policy on the Internet is implemented by an input rule at the distribution SOFAnode of SUPPLIER. This rule specifies "Application" as the bundle identification, "one year" as the time period and "push to distribution SOFAnode of CUSTOMER" as the action. The rule will be triggered every time a new version of APPLICATION appears at the distribution SOFAnode of SUPPLIER, and will push the APPLICATION bundle to CUSTOMER.

When the intranet of either a supplier or a customer consists of more than one SOFAnode, the distribution rules are also used for transporting bundles within the intranet. In case of a supplier, it is reasonable to expect that the development of a bundle will be done on a different SOFAnode than the one responsible for the interaction with customers. In case of a customer, it is similarly reasonable to expect that a bundle will be used on a different SOFAnode than the one responsible for the interaction with suppliers. In both cases, the distribution rules that describe an automatic transport of the bundles within the intranet are derived from the configuration of the intranet.

In our evolving example, an input rule at the development SOFAnode of SUPPLIER specifies "Application" as the bundle identification, "one year" as the time period and "push to distribution SOFAnode of SUPPLIER" as the action. The rule will be triggered every time a new version of APPLICATION is released at the development SOFAnode of SUPPLIER, and will make the Application bundle available for pushing to CUSTOMER.

3.3. Bundle Sharing on Intranets

Although the distribution rules can be used for the transport of bundles in an intranet, the mechanism suffers from two drawbacks. First, copying a bundle to all SOFAnodes in the intranet can waste resources, especially when the intranet is fast enough to transport the bundle on demand. Second, copying a bundle to all SOFAnodes in the intranet can violate the licensing model, especially when concurrent or as-a-book licensing is used. To avoid these drawbacks, we introduce a bundle sharing mechanism that complements the bundle distribution rules in an intranet.

Sharing of a bundle is initiated by a distribution rule that makes the bundle available for sharing. The SOFAnode that makes the bundle available for sharing acts as a *share manager* of the bundle. Other SOFAnodes within the intranet can become *share clients* by requesting the bundle from the share manager. The share manager is responsible for managing both the sharing and the licensing of the bundle. When the concurrent licensing is used, the share manager allows physical copies of the bundle to exist on all share clients that request it, limiting the number of copies that execute concurrently as necessary. When the as-a-book licensing is used, the share manager allows only a licensed number of physical copies of the bundle to exist on the share clients that request it, moving the bundles between share clients as necessary. To achieve this, the share manager relies on the SOFA environment, which can install and uninstall bundles on the share clients automatically, as well as notify the share manager about the execution of a bundle.

In our evolving example, an input rule at the distribution SOFAnode of CUSTOMER specifies "Application" as the bundle identification, "unlimited" as the time period and "share" as the action. The rule will be triggered every time a new version of APPLICATION appears at the distribution SOFAnode of CUSTOMER, and will make the Application bundle available for sharing within the intranet of CUSTOMER.

To take full advantage of the bundle sharing mechanism, SOFAnet has to complement it with a searching mechanism, so that potential share clients can discover available share managers. As a significant improvement, the SOFA component system can rely on the searching mechanism to locate components on demand when launching an application. The searching mechanism is similar to peer-to-peer networks in that it queries the share managers on the intranet directly, rather than relying on a centralized authority to keep track of the share managers and performing the search.

The searching mechanism can identify a bundle not only by its exact name and version, but also using its attributes and the attributes of its content. This allows the SOFA component system to launch an application only by specifying the interfaces or functions provided by the application, regardless of the specific supplier and implementation.

In our evolving example, launching APPLICATION on any execution SOFAnode of CUSTOMER will cause that

SOFAnode to search for the Application bundle. The search will discover the distribution SOFAnode of CUSTOMER, which acts as a share manager for the Application bundle. The share manager will consult the license object in the bundle and either allow or deny sharing, depending on the number of concurrently executing APPLICATIONS.

The overall structure of the SOFAnet, as described in this paper, is sketched on Figure 2. A working implementation of SOFAnet is available from our website [3].

4. Related Work

The design and implementation of SOFAnet is related to a number of ideas and technologies that became prominent recently. These include software packaging, software distribution and updating, push technologies and peer-to-peer networks.

An important representative of the related technologies is the OSGi service platform [10]. OSGi specifies a framework for delivering and executing software, typically in embedded devices. The specification introduces *bundles* and *services*. A bundle is a software distribution unit, also playing the role of an application, which can be installed, uninstalled, started, stopped and updated. A service is a long-running application available to other applications, which can be started, registered and looked up. OSGi, however, does not standardize distribution and licensing, which is left to the implementation. Oscar [6], as an example of an open-source OSGi implementation, introduces remote repositories that hold lists of available bundles, and can install bundles directly from their repositories.

A number of projects targets packaging of software in general. The more advanced packaging technologies augment bundles with information about deployment, configuration and dependencies of the packaged application. The packaging technologies are accompanied by package management systems and often also by update and upgrade services. The representatives of this category are the RPM [4][12] or DEB [2] packages and APT services [1]. These technologies allow downloading and updating bundles from the Internet repositories, and also help with installation, but do not cover the issues of distribution and licensing.

SOFAnet relies on pushing as one approach of transporting a bundle from a supplier to a customer. The pushing technologies experienced boom a decade ago in many forms, such as the PointCast Network. These technologies, however, distributed news rather than software. A customer could subscribe to a channel with a specific type of news. A supplier decided what news were pushed through the channel. Eventually, the pushing technologies disappeared, deemed too intrusive and overloading by the customers.

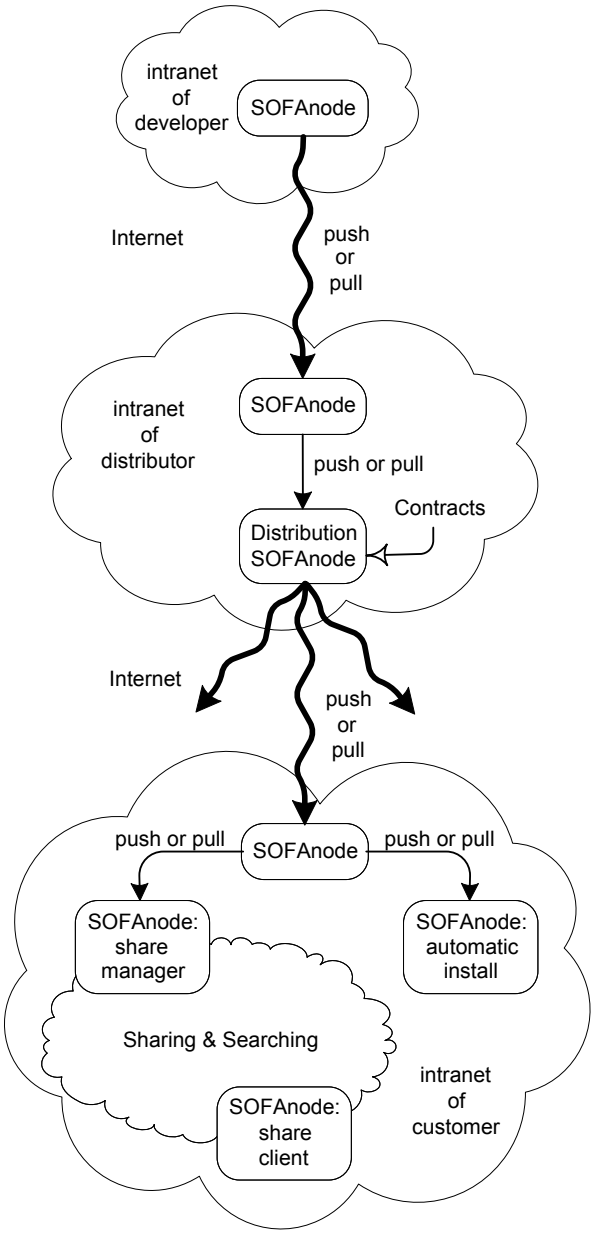

Figure 2. Overall structure of SOFAnet

The sharing mechanism of SOFAnet offers functionality similar to that of software protection systems such as FLEXnet [8]. These systems put emphasis on license enforcement and can complement SOFAnet.

The searching mechanism in SOFAnet is based on ideas pioneered by peer-to-peer networks. Contemporary peer-to-peer networks such as FastTrack [13] or Gnutella [5] usually offer two main features, searching for a bundle

and transporting a single bundle in many fragments from many suppliers. Typically, the peer-to-peer networks are designed to work in the relatively slow environment of the Internet. In SOFAnet, we take advantage of the comparatively fast environment of the intranets, where the searching is quicker and the transport of a single bundle from many suppliers can be replaced by sharing.

5. Conclusion

In the paper, we have singled out the distribution of software over the Internet as an application of the Internet that quickly grows in both popularity and usefulness. We have briefly explained why it is worthwhile to devise a ubiquitous distribution framework instead of the proprietary frameworks in place today, and what functions such a framework should offer.

We have described the design of SOFAnet as a ubiquitous distribution framework. SOFAnet provides support for distribution and licensing, including complex distribution and licensing models, in a form of an extensible framework. SOFAnet can distribute both complete software packages and incremental software updates in a manner that is seamlessly integrated into both the software and the Internet environment.

Although SOFAnet is a part of the SOFA component system, its principal features stay independent of SOFA. SOFAnet is in the stage of a working implementation, available from our website [3].

6. Acknowledgements

The authors would like to thank the members of the Distributed Systems Research Group for their work on the SOFA component system, which incorporates SOFAnet. The work was partially supported by the Grant Agency of the Czech Republic project number 102/03/0672.

7. References

[1] APT HOWTO Version 1.8.5. http://www.debian.org/doc/user-manuals#apt-howto.
[2] Debian GNU/Linux Version 3.0. http://www.debian.org.
[3] The Distributed Systems Research Group: The SOFA Project. http://sofa.objectweb.org.
[4] Ewing M., Troan E.: The RPM Packaging System. Proceedings of the First Conference on Freely Redistributable Software, Cambridge, MA, USA, February 1996.
[5] Gnutella. http://www.gnutella.com.
[6] Hall R. S. et al: Oscar, the Open Service Container Architecture Version 0.9.4a. http://oscar-osgi.sourceforge.net.
[7] Hnetynka P., Pise M.: Hand-written vs. MOF-based Metadata Repositories: The SOFA Experience. Proceedings of ECBS 2004, Brno, Czech Republic, IEEE CS, May 2004.
[8] Macrovision: FLEXnet Universal Licensing Platform. http://www.macrovision.com/products/flexnet.
[9] The Object Management Group: Licensing Service Specification Version 1.0. Document formal/00-06-17, April 2000.
[10] The OSGi Alliance: OSGi Service Platform Release 3. March 2003, http://www.osgi.org.
[11] Plasil F., Balek D., Janecek R.: SOFA/DCUP, Architecture for Component Trading and Dynamic Updating. Proceedings of ICCDS 1998, Annapolis, USA, IEEE CS, 1998.
[12] The RPM Package Manager Version 4.2. May 2003, http://www.rpm.org.
[13] Sherman Networks: Kazaa Version 2.6. November 2003, http://www.kazaa.com.