2012 International Conference on Information and Computer Applications (ICICA 2012)
IPCSIT vol. 24 (2012) © (2012) IACSIT Press, Singapore

The Research and Realization of Application Updating Online

Jiang Du[1,a], Ti Kang[1,b] and Yun Yang[2,c]

[1]Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing (400065), China

[2] Dept. of science &information system, Chongqing electric power corporation, 400014 Chongqing, China

Email: [a] clouddu@gmail.com, [b] clouddu@gmail.com, [c] yangyunly@163.com

Abstract. Software update plays a great supportive role in the software life cycle in maintenance, security, function additions and application environment expansion. Online update, which can reduce the cost and is convenient to the customer, is of great practical value. However, most research on software update is focused on a special software. So, a device with a prototype of the update program is brought out in this paper. The goal of the device is to update across platforms, reduce the artifacial participation in update, and increase the ability of reuse of this program.

Keywords: ACE, online update, cross platform, reusage

1. Introduction

Software update, as a great support for the development of software in the fields of self-robustness, security, application expansion, and so on, plays an essential part in the software life cycle. At present, the common update programs are all subject to a special software. For example, the QQ update program and windows update program. Although the 360 software housekeeper can inspect the version state of other softwares, it can only remind the user to call its own update program. So, a device of an universal update program for all types of application and system is put forward in this paper.

Before a new version releases, there are buges, lilmits in the software for the limits of the software design requirement and the application environment. As the increasement of the new needs, the modification and the new functions added are packed into a patch to release to the users. Software update is used to meet user's need better, improve the security of the software, expand the use range and offer more powerful functions.

Software update is divided into two types: system update and applicaiton software update.

1.1. System update

System update is used to change the old system into a new one with a higher version, take Windows Operation for example, users replaces WIN2000 for WIN98, replace WINXP for WIN2000, or download and install the patches as well. Usually, the update is performed easily through MS windows update software, and the new versions always have a better running environment with higher security and running speeds and so on.

1.2. Applicaiton software update

The most common way to update application software is to uninstall the old version, download and install the latest version. Software management programs, such as 360 software housekeeper, will remind and introduce users the latset version of the softwares to be updated. However, the stability and the practicability should also be taken into consideration besides updating, as not all the latest versions are the best appropriate, for most people, stability is much more important to function usually.

Moreover, as the wider use of Linux, more and more software are designed running across platforms, for this aim, this update program is based on the framwork of Adaptive Communication Environment (ACE). Then comes a brief introduction of ACE.

2. Adaptive Communication Environment

Adaptive Communication Environment (ACE)[1] is an open-source object-oriented (OO) framework which is free, with many core patterns for concurrent communication software. A rich set of reusable C++ wrapper facades and framework components is provided, which perform common communication software tasks across a range of OS platforms.

The main target of ACE is to provide high-performance and real-time communication services and applications on UNIX and Win32 for developers, the development of OO network applications and services are simplipied, which utilize communication between processes[2][3], event demultiplexing[4][5], explicit dynamic linking, and concurrency[7][8]. In addition, ACE automates system configuration and reconfiguration by dynamically linking services[9][10] into applications at run-time and executing these services in one or more processes or threads.

The main benefits of using ACE include[1]:

- Increased portability – it is easy to write concurrent networked applications on one OS platform and quickly port them to many other OS platforms with the help of ACE components.
- Better software quality – The design of ACE uses many key patterns that increase key qualities, such as flexibility, extensibility, reusability, and modularity, of communication software.
- Better efficiency and predictability -- ACE is carefully designed to support a wide range of application quality of service (QoS) requirements, including low latency for delay-sensitive applications, high performance for bandwidth-intensive applications, and predictability for real-time applications.
- Easier transition to standard higher-level middleware -- ACE provides the reusable components and patterns used in The ACE ORB (TAO), which is an open-source standard-compliant implementation of CORBA that's optimized for high-performance and real-time systems. Thus, ACE and TAO are designed to work well together in order to provide comprehensive middleware solutions.

3. Update Procedure

3.1. Environment instruction

The mode of this program is C/S. On the client side saved the main program and the update program, and all the update tasks are implemented by the update process, together with a config XML file to preserve the current version information of the main program. On the server side, there is also a config XML file too, in which the detail information of the update package is recorded. Together with update package, in which the files with latest version are included. The tomcat is installed in the server side, and the config XML file and the update package are defaultly placed in the ROOT directory.

3.2. Update Procedure

1) Input the new version code and the type code through command line to activate the update program;
2) Load update.xml, the local config file, to get the recent version id and type. You can also get the update state of the last update, which is successful or not;
3) Compare the new version id and type with the old, if the new type is different or the new version id is much bigger, then a update is need; or process stop;
4) If a update is need, download serverconfig.xml from the server. Then analyze this XML to get the information of the update package, such as the download path, the save path, the MD5 code and the execute script command when the package unzipped. The MD5 code will be saved to the update.xml;
5) Compare the new MD5 code with the last MD5 code, if different, it means that the package is new and need to be downloaded. Otherwise, the package is used and old, misused in a new update, and process stop;
6) Download the package when it is a new one according to the download path get form the serverconfig.xml, and save it to the appointed place;

7) Check the MD5 checksum of the recent update package, make sure that there is no error during the download;
8) Unzip the package when the package is complete and correct, execute the script commands to update the main program;
9) Stop the update program when update is finished.

4. Key Steps in Updat

4.1. Match version information

First, detect the latest version information of the main software. On the client end, when the main program detect new version information, it activate the update program and transfers the version code and the type code to it. A XML config file named Update.xml is kept in the client file, which is used to record the latest version information of the main program, including version code and the type code.

The basic information in this file is the IP, port number and server config file url. When a successful update is done, more information is recorded here, including the current version code and type code the update package name, download url, MD5 check code, the save path of update package and the state of update. Update.xml is as follows:

```
<?xml version="1.0" encoding="GBK" ?>
<config><server ip="10.10.10.66" port="8080" updateUrl = "/Serverupdate.xml"/>
<version command="./file/lua_copy.lua" type="2" versionId="1213">
<file MD5="b9c9dd85282dce7855ea234f8e9183a0"> update1.zip
</file></version>
<update state="2"><updateFile MD5 = "b9c9dd85282dce7855ea234f8e9183a0"
downLoadUrl="/update1.zip"  savePath="/update1" state="1">update1.zip
</updateFile></update></config>
```

As the update program stimulated, it will compare the current version code and type code with the new get ones. If a new version is defined, the update program whill get into update state; otherwise, it will be stopped.

4.2. Download process

When the update program get into update state, it will perform download process. In the download process, there are two main stelps for downloading, one is downloading the server config XML file, the other is download the update package.

1) Download the server config XML file

The url of the config XML file is got by the key word “updateUrl” from the Update.xml file. Download it and locate on the server named Serverupdate.xml, in which the detail information about the new update package is laid, including the version code, type code, the server save path, the download target path and the script command. The content is as follows:

```
<?xml version="1.0" encoding="GBK" ?>
<config><version type="2" versionId="1213"  command = "lua_copy.lua">
<version type="1" versionId="2"  command = ".\update\lua_copy.lua">
<fileMD5="xx11" downloadUrl= "/ikeeper2008/sound/test.zip" savePath =". "> test.zip </file></version>
</config>
```

The Http protocol is used in the download transmission. When this file is saved to the target location successfully, it will be renamed as serverconfig.xml .

2) Download the update package

The second step is downloading update package. The download url is gained by matching the new version code and the type code got from the main program with the ones lying in the serverupdate.xml one by one. When it matches successfully, the url is picked up by the keyword "downloadUrl", and is written into a HTTP command, as the downloading process is so done by HTTP protocol[10]. The details are as follows:

```
sprintf(buff, "GET %s HTTP/ 1.1 \r\n Host: %s \r\n Connection: Keep-Alive\r\n\r\n", get_file.c_str(), server_ip_.c_str());
connet_handle_->peer().send(buff, strlen(buff));
```

```
while(!connet_handle_->recv_finish() && !connet_handle_->down_load_error() && !stop_)ACE_OS::sleep(1);
down_resulte=(connet_handle_->recv_finish()&&!connet_handle_-> down_load_error());
connet_handle_->close_file();
```

The update package is saved to the target location get from the keyword "savePath" indicated in the serverconfig.xml .

4.3. Check process

For the assurence of the security and the integrality of downloading, the check process done by MD5 checksum[11] is performed after the download process, which check up the correctness of the transmitted data received by a hash operation. Before the transmission, a hash code of this update package is got from the serverconfig.xml, on the server. When it is downloaded, a new hash code will be calculate out, if it is equal to the previous one, then this package is transmitted successfully; otherwise, there are errors during transmiting. The check process in detail is as follows:

```
md5_test.MD5File(file_name);
unsigned char buff[17] = {0};  char code[34] = {0};
char *q = code;  md5_test.MD5Final(buff);
for (int i = 0; i != 17; ++i){
      unsigned char x = buff[i];
      char temp[5];
      ACE_OS::itoa(x, temp, 16);
      sprintf(q, "%s", temp);
      q += strlen(temp);
}
return (ACE_OS::memcmp(code, md5, 32) == 0);
```

4.4. Decompress process

The update package is to be decompressed after MD5 check procession. The default fomat of the compress file for this update progrom is zip, since it is used universally not only in Windows, but also in Linux, which make this update program more portable and universal. The decompress process in detail is as follows:

4.5. Perform concreate update operation

All the concrete update operations are written in a scrip file to avoid unnecessary modifications in each update process, since there are differences between each concrete update operations. The functions of the update program are expanded greatly with out any modification on the main framework of the program, only need to change the script. Here, Lua is adopt in this program, for it is written by C-like language, is of great efficiency and associativity with C/C++, and running across platforms. Since only the config files and the script need to be rewrote, the security and maintenance are increased.

5. Summaries

This update program writes in C++, using ACE framwork, efficient and universal, can perform successfully on both Winxp and Linux. All the config information and concrete update steps are separated form the update program, which increased the reusability, maintenance and security greatly. The whole program including the code above has been compiled successfully on Visual Studio 2005 and Gcc-4.1.0-3. As the development of network applications, the online update will become more and more universal, modularized in functions and intelligent, easier, faster and safer.

6. Acknowledgment

The work presented in this paper was supported by Research on Policy of Security Audit for Application Services in Intranet project.

7. References

[1] Douglas C. Schmidt, etc. ADAPTIVE Communication Environment Technical Documentation [M]. 2002. http://www.cs.wustl.edu/~schmidt/ACE.html

[2] D. C. Schmidt, T. H. Harrison, and E. Al-Shaer, “Object-Oriented Components for High-speed Network Programming,” in Proceedings of the 1st Conference on Object-Oriented Technologies and Systems, (Monterey, CA), USENIX, June 1995.

[3] D. C. Schmidt, “IPC SAP: An Object-Oriented Interface to Interprocess Communication Services,” C++ Report, vol. 4, November/December 1992.

[4] D. C. Schmidt, “The Reactor: An Object-Oriented Interface for Event-Driven UNIX I/O Multiplexing (Part 1 of 2),” C++ Report, vol. 5, February 1993.

[5] D. C. Schmidt, “The Object-Oriented Design and Implementation of the Reactor: A C++ Wrapper for UNIX I/O Multiplexing (Part 2 of 2),” C++ Report, vol. 5, September 1993.

[6] D. C. Schmidt, “An OO Encapsulation of Lightweight OS Concurrency Mechanisms in the ACE Toolkit,” Tech. Rep. WUCS-95-31, Washington University, St. Louis, September 1995.

[7] D. C. Schmidt and C. D. Cranor, “Half-Sync/Half-Async: an Architectural Pattern for Efficient and Well-structured Concurrent I/O,” in Pattern Languages of Program Design (J. O. Coplien, J. Vlissides, and N. Kerth, eds.), Reading, MA: Addison-Wesley, 1996.

[8] D. C. Schmidt and T. Suda, “An Object-Oriented Framework for Dynamically Configuring Extensible Distributed Communication Systems,” IEE/BCS Distributed Systems Engineering Journal (Special Issue on Configurable Distributed Systems), vol. 2, pp. 280–293, December 1994.

[9] P. Jain and D. C. Schmidt, “Service Configurator: APattern for Dynamic Configuration and Reconfiguration of Communication Services,” in The 3rd Pattern Languagesof Programming Conference (Washington University technical report #WUCS-97- 07), February 1997.

[10] R. Fielding，UC Irvine，J. Gettys，etc. Hypertext Transfer Protocol -- HTTP/1.1[M]. Network Working Group，Request for Comments: 2616, 1999

[11] R.Rivest. RFC 1321 - The MD5 Message-Digest Algorithm[M]. MIT Laboratory for Computer Science and RSA Data Security Inc. April 1992.