A CGI+AJAX+SVG Based Monitoring Method for Distributed and Embedded System

Wu Zhangjin, Huang Xingwen, Ding Ying, Zhou Rui, Zhou Qingguo*
Distributed & Embedded System Lab,
School of Information Science & Engineering, Lanzhou University, China
{zhangjinw,hxwsimon,dingy09,ethanchou}@gmail.com, zhouqg@lzu.edu.cn

Abstract

This work focuses on the integration of CGI, AJAX and SVG technologies into the B/P/S architecture for monitoring the distributed and embedded Linux system. The traditional monitoring technologies and architectures do not sufficiently satisfy either the transparency of the distributed or the resource limitation of the embedded at the same time, but these two aspects are basic for distributed and embedded system. This paper describes the design decisions on integrating CGI, AJAX and SVG into the B/P/S architecture, analyzes the shortcoming of the traditional monitoring methods, introduces a sample implementation and points out some potential application domains and ideas for future development.

Keywords: Distributed, Embedded, Linux, CGI, AJAX, SVG, B/S, Proxy, Monitoring.

1. Introduction

As the embedded systems used more and more widely in our daily life, we are stepping into the age of pervasive (or ubiquitous) computing[4] : the computers are ubiquitous and pervade various aspects of our life. Some computers are not like the common computers for they have limited hardwares and softwares, such as digital machine tool, elevator, electronic dictionary, etc. If they are connected together with a complex topology structure to make up a distributed system[3] , a problem will stand up ahead - how to monitor it?

By analyzing the shortcoming of the current monitoring methods[7] on the special requirements (transparency, resource limitation, etc.) of the distributed[6] and embedded system[8] , this paper will illustrate why and how CGI, AJAX and SVG technologies integrated into the B/P/S architecture. At last, a sample will be introduced.

CGI (Common Gateway Interface)[1] is a way of creating dynamical web pages, through which, the client can monitor the states of the controlled embedded devices distributed somewhere. As a standard interface between an external application (gateway) and information server (HTTP), there is no need to build a special client but a common web browser and transform C/S architecture to B/S architecture to implement two ways of communication, which means that in a distributed and embedded system, the server nodes can show and change their internal state on a client's request.

AJAX (Asynchronous Javascript and XML)[2] is a combination of technologies aiming at providing an improved user/application interactivity in the context of web-based service-oriented computing. With the help of XML, XSL and CSS[5], AJAX supports MVC by nature for using XML as the data presentation format, XSL and CSS as the data display format. Based on the XMLHttpRequest API and the SetTimeOut function of Javascript, AJAX provides a method for handling service invocations asynchronously while interacting with other applications/users in the foreground.

SVG (Scalable Vector Graphics) [10] is a language for describing two-dimensional graphics and graphical applications in XML. Hence, the multi-media output can be easily generated for it has been supported by the common browsers (e. g. firefox 3.0, IE 7.0) without installing extra plug-ins.

B/P/S architecture here is based on the B/S architecture and a proxy (agent) added in between the Server nodes and web browsers for hiding the implementation details and providing a universal interface. At the same time, by integrating the above three technologies, the user/application interactivity, module scalability, interfaces

978 -1-4244-1866-4/08/$25.00 © 2008IEEE

standardization and display diversify will be easily implemented.

The following section analyzes the requirement of the distributed and embedded Linux system and the deficiency of the current technologies and architectures. After that, the integration of CGI, AJAX and SVG into the B/P/S architecture will be explained, afterward, an implementation sample will be introduced for further illustration. At last, a conclusion is given and some ideas for potential application domains and future work are pointed out.

2. Shortcoming of the Traditional Monitoring Methods

There are lots of monitoring methods for distributed and embedded system with various technologies and architectures. Let's show some of them in the following table.

Technology	Architecture	Distributed	Embedded
HW-based	CLI&GUI	0	0
IP-based	C/S	1	1
WEB-based	B/S	2	2

Table 1. *Current Monitoring Methods*

The **Distributed** and **Embedded** columns describe whether the related technologies and architectures is suitable or not. **2** means YES, **0** means NO, and **1** means not completely.

HW-based means hardwares directly connected and perhaps just provide Command Line Interface or Graphical User Interface via a displayer, Hence, it won't be suitable for the embedded for lacking a displayer and not acceptable to the distributed for lacking the scalability.

IP-based denotes the systems using Client/Server architecture with the TCP/IP or UDP protocol stack(e. g. ssh/sshd, telnet/telnetd). They are mostly point-to-point, thereby, not good enough or will be difficultly used to the distributed system. At the same time, it will not be easy to generate an effective visual display.

WEB-based mainly due to the modern network technologies, including hardware and software parts. The latter which we concerns embraces lots of network protocols, algorithms, standards for data presentation, transmission, layering, sharing and user-friendly etc.

Current mainline web technologies including CGI, Java, ASP, .Net, AJAX, PHP, Python, Ruby, SVG are web-based. They may apply to the distributed system, but after considering the expense (such payment of royalty) and the resource limitation of embedded system, perhaps CGI, Java, PHP, AJAX, SVG will be left. For further resource limitation, PHP will need an independent interpreter. If using GD2, more storage will be needed. What about Java Applet, a non-lightweight enough JVM plug-in should be installed with the browser. At last, only CGI, AJAX and SVG will be left.

CGI can be programmed in shell language(or c language) to generate the dynamical HTML web page on the support of the Bash integrated into the embedded Linux system[9] ,hence, no extra interpreter needed but a lightweight web server. Considering the user/application interactivity, AJAX provides asynchronous communication with XMLHttpRequest object and SetTimeout function to implement granular updating. If scalability and openness considered, AJAX gives us the uniform data presentation language, XML. What about user-friendly, SVG produces diversity output and multi-media display to enhance user's experience.

As mentioned above, that is why to combine CGI, AJAX and SVG together, but it does not fit the transparency of the distributed system either. Hence, a new architecture should be designed, which is a proxy (or agent) should be added to the B/S architecture for providing an uniform monitoring interface and hiding the details of the behind. Such an architecture and the integration of the above three technologies into it will be described in the next section.

3. Integration CGI,AJAX,SVG into the B/P/S Architecture

After importing the Proxy(or agent) and integrating the CGI, AJAX and SVG into the B/S architecture, it turns into the B/P/S architecture as figure 1 shows below.

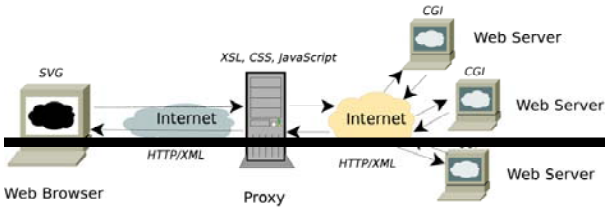


Figure 1. *B/P/S Architecture Overview*

In such an architecture, there are basically three modules:

- Web Browser, requests services
- Web Server, executes services
- Proxy(or agent), works as either a web browser or a web server to execute and request services for responding the browser and requesting the services from an indicated web server hiding from the Browser and behind the proxy.

Communication between them are based on the exchange of XML (Extensible Markup Language,provided by AJAX) document, Transportation of these documents, the HTTP (Hypertext Transfer Protocol) protocol.

The basic communication and transportation procedure may be like this:

1. Administrator send a request via the web Browser to the Proxy, the Proxy return the main administration interface, including the XML document and the XSL (The Extensible Stylesheet Language), CSS(Cascading Style Sheets), JavaScript files.
2. The web Browser interpret the XML document with the XSL, CSS file to HTML/SVG document and display them. At the same time, the XSL, CSS for displaying the XML documents will be stored in the local cache and won't need to request again, so other data exchange between the three modules will only left XML documents, Hence, the data exchange will speed up. Beyond that, if the Javascript file including the SetTimeOut function and the XMLHttpRequest object, the timed granular page updating will be executed by the browser itself which means that a dynamical page updating is implemented.
3. The Administrator send a request to an indicated web Server(the server node in the target distributed embedded system), The Proxy(exactly, the web server built on proxy) response this request immediately and forward the request to the indicated web Server via the built-in HTTP client on the Proxy. after that, the Server response this request, generate a dynamical XML document via executing CGI program to collect the relative info from the server node on an embedded device(e. g. info from external sensor, system info). afterward, the XML document will be sent to the Proxy and finally to the Browser. The Browser handle it as the second step shows.

The Administrator sometimes may just send a request and not indicate the exact web Server(this is a very important requirement of the distributed system, transparency), At this time, the Proxy should know how to forward the browser's request to one or several server nodes. To implement it, an embedded database system is needed to store the information of the server nodes, and some pre-designed load balance algorithms should be implemented on the Proxy.

To demonstrate the above procedure clearly, figure 2 have been drawn as follow.

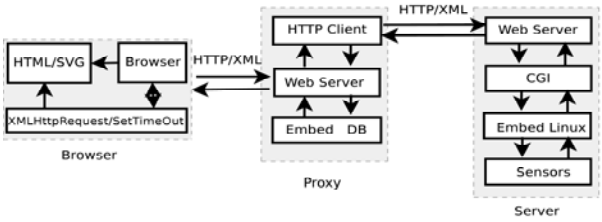
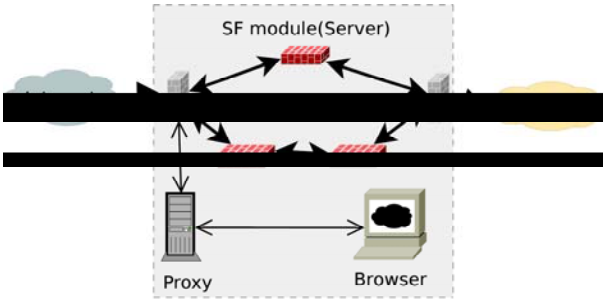


Figure 2. *Functional Blocks*

4. Implementation Example: Scalable Firewall(SF)

Scalable Firewall[11] is a firewall, based on Netfilter/Iptables and Linux bridge technologies. It implements both the hardware scalability(modular and plug-in-out-able) and software expandability. The connection of the hardware modules(server node) can be organized as the serial, parallel topology structure or their combination. The figure 3 shows the possible connection modes with the monitoring method discussed in this paper.



method this paper described, a Proxy with the Browser has been imported in. At the same time, a load balance algorithm based on the MAX number of rules(of every "SF module") has been implemented on the Proxy. Some standard CGI interfaces have been designed on the SF modules, as a result, an universal administration interface has been built to reduce the complexity of the administration and enhance the user-friendly of the monitoring result.

Figure 4 shows a monitoring result with the SVG graph[12].

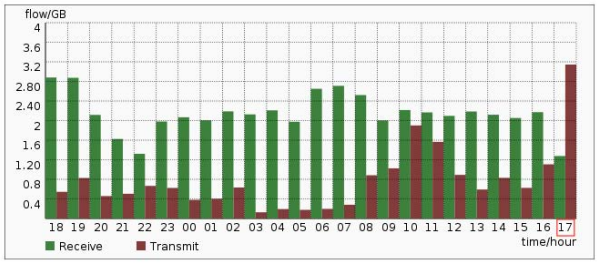


Figure 4. *The SVG output of Scalable Firewall*

5. Potential Application Domains

As the distributed and embedded system developing, a good monitoring method becomes more and more important. lots of application domains may need such a method we have discussed here, such as:

- Network performance of the firewalls[111 , routers, bridges monitoring
- Temperature and humidity[1] of the greenhouse,laboratory monitoring
- Gas concentrations of the underground mine monitoring
- Traffic flow rate of the highroad, airways monitoring

Figure 3. *The Architecture of Scalable Firewall*

The red blocks in the above figure present modules of the Scalable Firewall, named "SF module", all of which work together to implement the function of a whole firewall system, the gray ones are the abstract of “in” and “out” firewall modules for connecting the internet beside.

Such a system has a very complex topology structure, but with the help of the monitoring

- Noise of the recording studio monitoring

6. Conclusion And Future Work

In this work, the principle and a basic implementation sample of the monitoring method based on the CGI, AJAX and SVG technologies with the B/P/S architecture are presented. The method is applied to the distributed and embedded system for it achieves the transparency of the distributed and meets resource limitation of the embedded. At the same time, it can provide a user-friendly output and reduce the work load of the developers.

For future work, some standard XML documents and SVG graphic patterns may be designed for the corresponding application domains. In addition to this, some shell CGI and XSL libraries can be implemented, last but not least, the universal interface on the Proxy module should be improved to a stable framework.

References

[1] Grisha Spasov, Nikolay Kakanakov, 2004, CGI-based applications for distributed embedded systems for monitoring temperature and humidity, CompSysTech’2004.

[2] Ramesh Bharadwaj, Ramesh Bharadwaj, 2006, Functional "AJAX" in Secure Synchronous Programming, WWW2006.

[3] Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya, 2006, Distributed Systems and Recent Innovations: Challenges and Benefits, InfoNet06.

[4] ROBERT GRIMM etc, Programming for Pervasive Computing Environments, InfoNet06.

[5] Hong-Taek Ju etc, An Embedded Web Server Architecture for XML-Based Network Management.

[6] Scarlet Schwiderski, April 1996, Monitoring the Behavior of Distributed Systems.

[7] George Gousios,Diomidis Spinellis, May 2002, A Comparison of Portable Dynamic Web Content Technologies for the Apache Server, 3rd International System Administration and Networking Conference Proceedings, pp.103–119.

[8] Nasser Jazdi, Component-based and Distributed Web Application for Embedded Systems.

[9] Karim Yaghmour,April 2003, Building Embedded Linux Systems, O'Reilly, 0-596-00222-X.

[10] SVG, 2008, URL: http://www.w3.org/Graphics/SVG/

[11] Scalable Firewall, 2007, URL: http://elf.oss.lzu.edu.cn/

[12] vnStatsvg, 2008, URL: http://vnstatsvg.sourceforge.net/