THE ASTERISK AS A PLATFORM FOR THE DEVELOPMENT OF TELECOMMUNICATION SERVICES.

Yousef Daradkeh[(1)], Andrei Ustinov[(2)], Dmitry Namiot[(2)] ,
(1) Department of Electrical and Computer Engineering,
University of Calgary, 2500 University Drive NW,
Calgary, Alberta, Canada T2N 1N4
(2) Computational Mathematics and Cybernetics Faculty,
Lomonosov Moscow State University,
Moscow 119899, Russian Federation
daradkehy@yahoo.ca, dnamiot@gmail.com

ABSTRACT

We would like to share with you one interesting model for the use of the PBX Asterisk. To be more precise, the main idea is to use the Asterisk as a platform for the development of telecommunication services. With its open source software and API, the Asterisk can bring the development of telecommunication services down to a simpler process of Web programming thus considerably lowering "the entrance barrier" for those involving in the programming of new services.

KEY WORDS
Advanced Development of Telecommunication web service, principal component analysis, Proxy services, Java script.

1. Introduction

We propose to integrate a new component (proxy) into the Asterisk platform. The main functionality of the proxy is to translate telecommunication calls into HTTP requests to external web services. Telecommunication services are located separately from the PBX, while the information they receive from Asterisk is presented as a HTTP-request. Technically, HTTP GET/POST request is a request, in which external telecommunications service passed information about the subscriber's name – CallerId Name, caller's number – CallerId Number and called number - Extension. Upon receiving necessary parameters, such as (calling/called number) a web service produces and forwards its instructions to the proxy. The latter receives and translates them into Asterisk instructions. The development of such services under the architecture described above is similar to a conventional CGI-script, for which there is a plenty of programming tools. As a result a programmer doesn't need to be familiar with the Asterisk API.

Asterisk is both an open source toolkit for telephony applications and a full-featured call-processing server in itself. It can be a standalone system, or used as an adjunct to a previously existing PBX or Voice over IP (VoIP) implementation. It can be software only, moving calls around via IP, or it can have a variety of hardware interfaces to directly tie in with existing TDM (Time Division Multiplexing) equipment. Asterisk is not a VoIP platform; it is a Computerized Telephony Integration platform, which just happens to have a number of very useful input/output channels through VoIP. Asterisk can just as easily be a server that has no Internet connectivity, but uses PCI-card-based analog or digital trunks to process calls -- an important distinction between Asterisk and many other systems.

Until now, Asterisk systems treated mostly as a software replacement for the traditional PBX.
And we would like to offer a new point of view – Asterisk system as a platform for the telecom services. In this approach we do not need always replace the traditional PBX, we can use Asterisk as a supplemental system. The goal for this add-on is just to perform a processing for the calls. But it is actually only one part of the story.
Let's consider in a more detailed way the model for the development of telecommunication services we are offering. In the many aspects it is new model merges web and telecom programming. The model explains how to use web development right for the telecom services. At the first time this model was introduced by the company AbavaNet (D. Namiot, M. Scheps-Schneppe). At that time the model was practically implemented for INAP protocol-based service nodes in Audiotele Company. The next implementation was based on CSTA protocol for SI-2000 from Iskratel. And this article describes the model implementation for Asterisk platform. It was implemented first time by A.Ustinov, as a part of his post-graduate qualification paper in Moscow State University.

2. Web Service

This new approach introduces the proxy-based lightweight framework for providing the secure access to the Web services being requested by the clients. The basic idea is to deploy a proxy Web service that receives the request from the end client on behalf of the actual Web service. The proxy service authenticates the end client by validating the client's credentials, which he/she had sent

Along with the Web service request. If the client is authenticated successfully, he/she will be given access to the requested service.

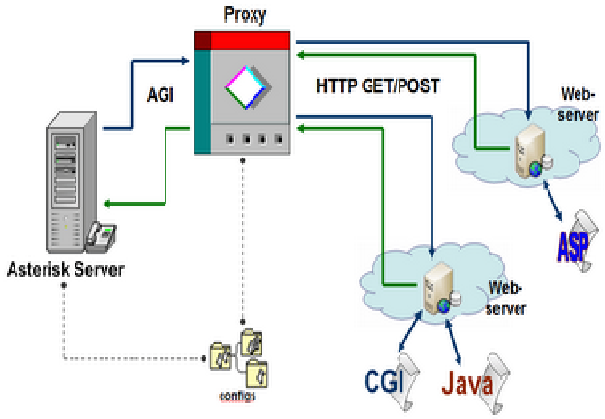

Figure 1: Its global framework is shown below.

As seen the AGI-proxy component is at the heart of the model. The AGI-proxy is a Java-based application implemented on the basis of FastAGI, an open source library. The AGI-proxy is installed directly on the PBX Asterisk side.
In fact, the Asterisk represents the same thing for the AGI-proxy, as the J2EE container does for a Java-servlet. All calling massages produced by the Asterisk come to the proxy within the method service with two parameters: interfaces AgiRequest and AgiChannel. Through the first parameter one can get the information about the calls (caller name / number, called number, context of the call, feed settings, etc.) and through the second one the interaction with the Asterisk is carried out (call termination, transfer mode and etc.). Subscriber's name (CallerIdName), caller's number (CallerIdNumber) and called number (Extension) are wrapped within a string-parameter by the AGI-proxy. Then the AGI-proxy executes an HTTP request on its behalf to an external web service, whose URL is set in the configuration. The response from the invoked service is seen as an indication to what to do with the call. The call may be terminated or redirected to a specified number; a media file may be played. The instructions returned by an external service must be presented in an xml-format. It has a very simple scheme. For example:

<method> - this section specify a method
<method-name> - describes a name for the method
<method-param> - this section describes a parameter
<param-name> - describes a name for the parameter
<param-value> - describes a value for the parameter
<param-type> - describes a type for the parameter

An example of the typical response of a web service is shown below:

```
<xml version="1.0" encoding="UTF-8" ?>
<methods>
 <method>
  <method-name>playMediaFile</method-name>
  <method-param>
   <param-name>MediaFile</param-name>
   <param-type>String</param-type>
   <param-value>demo</param-value>
  </method-param>
 </method>
</methods>
```

On receiving this xml-response, the proxy will play the media file "demo" (playMadiaFile).

And the following figure displays call sequence in our approach:

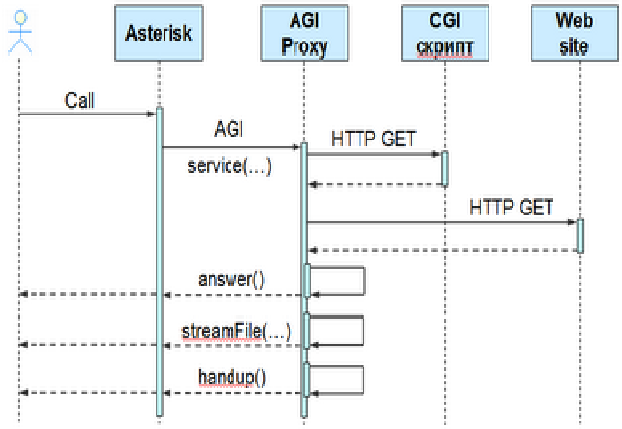

Figure.2: architecture the sequence of calls is reduced to a very simple diagram.

As you see at the first hand the incoming call is accepted by Asterisk (variant: by the legacy PBX, forwards the calls to Asterisk). Asterisk terminates the incoming call on our proxy. As the result of incoming call this proxy distributes the ordinary HTTP request down to the service. And the response from this service (yes, just

Ordinary HTTP response – that is an idea!) dictates what we have to do with the source call.

We would like to say a few words about external web services and provide you with one simplified example. Web services within this model may have many different purposes - from the statistics (to record a call and terminate it - in voting systems) to the Asterisk dialplan control (auto- answering and call forwarding in communicationsystems).
To demonstrate the functionality the proxy we need two files agiproxy.jar, AgiProxyScript.java and two configuration files agiproxy.conf, fastagi-mapping.properties. The files should be copied to the directory /var/lib/asterisk/agi-bin/. Let's make the following steps to configure the proxy:

1. Set the extensions (calls), to which the proxy will "react". In the file /etc/asterisk/extensions.conf one may write, for example, the following instruction: exten => _1XXX,1,Agi(agi://localhost/proxy.agi) (To run the proxy when a four-digit number beginning with "1" is dialing up)

2. Set the URL of an external service (see below), with which the proxy will interact. For example: urltunnel=http://localhost:8080/agiweb/agi.jsp
(to perform an HTTP-request to the agi.jsp at the localhost)

3. Set parameters for the interaction of the proxy with the Asterisk:
hostname=localhost (Asterisk server name) username=manager (user name indicated in manager.conf) password=pa55w0rd (password from manager.conf)

4. Compile AgiProxyScript.java for your operating system as follows: to run the instruction written below from /var/lib/asterisk/agi-bin/-directory (for Java version 1.6): Java c -cp.: asterisk-java-0.3.1.jar:agiproxy.jar AgiProxyScript.java

5. Run the Asterisk, using, for example, the following instruction:asterisk-vvvvvcd

6. Launch the AgiServer by invoking the instruction below from /var/lib/asterisk/agi-bin/ -directory: java cpagi proxy.jar:asterisk-java0.3.1.jar:.org.asteriskjava.fastagi. DefaultAgiServer

as the example of external services a trivial java server page (agi.jsp) has been written. It parses the string of Parameter received from the AGI-proxy and determines a called number (Extension). Then it creates instructions for Asterisk depending of the called number: for the number "1200" a media file will be played, a call having the number "1300" will be redirected to the number "500", any other numbers will be terminated.

In conclusion, let us remind you the advantages of using this model to develop telecommunication services as opposed to the traditional "programming" dialplan in configuration files:
1. The entry barrier for programmers of new telecommunication services becomes lower.
2. The possibility to develop services using various technologies, such as CGI, JSP, ASP.NET.
3. The possibility of integration of new external services without any changes on the PBX side.

Conclusion

This article describes a basic idea for the new approach to the telecom services development. This new approach converts the original task right into web development and does not require from the developers the telecom expertise. We think this approach has got a big potential due to considerably lowering "the entrance barrier" for those involving in the programming of new telecom services.

About the authors

Dr. Yousef Daradkeh is Associate Researcher at the Department of Electrical and Computer Engineering at the Faculty of Schulich School of Engineering, The University of Calgary (Canada).

A. Ustinov (at the time of writing) juts finished his post-graduate courses in Moscow State University.

Professor Dr. Dmitry Namiot is Senior Researcher of open Information Systems laboratory at the faculty of Computational Mathematics and Cybernetic, The Moscow State University (Lomonosov).

References

[1] http://www.asterisk.org/
[2] http://www.easyasterisk.org/
[3] http://updates.zdnet.com/tags/asterisk.html
[4] http://ieeexplore.ieee.org
[5] http://asterisk-java.org/development/tutorial.html
[6] http://asterisk-java.org/development/apidocs/index.html
[7] http://openfor.blogspot.com
[8] http://abava.blogspot.com