Open Source Telecom Gateway for Home Automation

Dmitry Namiot Moscow State University, Moscow, Russia dnamiot@gmail.com
Manfred Sneps-Sneppe University of Latvia, Institute of Mathematics and Computer Science, Latvia manfreds.sneps@gmail.com

***Abstract* – This paper describes a practical approach for using telecom gateway in Home Automation. Our article discusses the tasks and possible solutions, describes the places for telecom services within the Home Automation projects. Also we introduce a new approach for developing telecom services with Asterisk and provide Open Source implementation for this idea. The described package has been used on practice for Smart Home project on Latvia.**

INTRODUCTION

This paper describes our vision for using telecom services within the automation projects. It is based on our experience in telecom area and practice we've got during the implementation of Smart Home automation projects.

At the first hand let us clearly set the basic points for our approach. At the first hand, we do believe into mashups only. In other words all our past experience says that there is no way for one system to support all of the tasks. No size fits all. All the systems does not matter how they were planning at the beginning finished as the some cooperation of the services. The only successful projects were the projects originally oriented (and developed as) to orchestration of the services. At the second we do believe that time to market is the most important factor during the development. It means that we need to provide (to choose) the development tools most developers are familiar with. There is almost no way to convenience the developers start to lean completely new API's or development environments. At least it is too complex task to be solved during the one project. We need something that is simple, has got quick learning cycle and could be used for the fast prototyping too.

Now let us describe the place for the telecom services within the Home Automation projects. We can highlight two possible areas: voice responses - e.g. using text to speech service and explain measured data by voice as an answer to user's call and opposite task – using call as a switch on/switch off command for our equipment. From our vision it covers all the areas (the rest is simply a combination of two methods).

As the next step let us describe our vision to computer telephony and appropriate services. Technically we are speaking of course about one simple thing – the ability to catch the call and process it programmatically. Actually, it is all. Let us see what can we do next. By our vision any (ok, almost any, to be polite) telecom service could be presented as a combination of the finite set of basic services. Let us count them:

1) application accepts a call and hangup. For example all the voting services (or switch in/switch off) services look so. Just get a call (get A-number), do some action on the server side and drop a call;
2) call redirection. Applications accepts a call, get a new number by the own (e.g. some database lookup) and redirects the incoming call to the new destination;
3) media play. Just got a call and play some media file (static or dynamic) in the answer
4) record media file. Just got a call and write the voice to some file
5) DTMF recognition. Just get and recognize tone signals from the line

On practice, it is all do we need. The vast majority of the services are actually just a combination of the above 5. It is not a big list actually, so we may expect that we will be able to pickup an appropriate development tools for them.

BASIC TOOL

As a basic telecom-enablement tool for the Home automation projects we choose Asterisk.

Asterisk is software that turns an ordinary computer into a voice communications server. Asterisk powers IP PBX systems, VoIP gateways, conference servers and more. It is used by small businesses, large businesses, call centers, carriers and governments worldwide. Asterisk is free and open source.

As per official site Asterisk is often referred to as "the open source PBX" and it's true that you can use Asterisk to build a PBX. But a PBX is only one of many applications you can build with Asterisk. Asterisk also is gateway, voice mail, IVR etc.

One big advantage for the production is the fact that Asterisk is quite popular, so it is not a niche solution used with one or two projects only.

Asterisk has got open API too, so we can actually program our own applications on it. And Asterisk's API is enough to implement the above mentioned basic services as well as the combine them according our need.

But of course this API is just yet another API for the developers. And it has got own price (complexity). So for example Java developers we are traditionally working with must study new things for programming Asterisk.

E.g. the easiest way to interact with Asterisk from Java applications is via the FastAGI protocol. AGI scripts can handle either incoming calls or calls originated via the Manager API

The AGI (Asterisk Gateway Interface) facility allows you to launch scripts, from the Asterisk dial plan. Traditionally communication between the scripts and Asterisk was via standard input and standard output and scripts had to run on the same machine as Asterisk. Due to the large amount of time a Java Virtual Machine needs for startup and the discomfort of having to install a Java environment on the PBX box(es) Java has not been the language of choice for writing AGI scripts.

These drawbacks have been addressed by the addition of FastAGI to Asterisk. FastAGI is basically AGI over TCP/IP socket connections instead of using standard input and standard output as communication medium.

Using FastAGI you can run a Java application (on the same machine that runs Asterisk or on a separate machine) that is only started once and serves AGI scripts until it is shut down. Combined with Java's multithreading support you can build pretty fast AGI scripts using this protocol.

Asterisk-Java helps you with running your Java based AGI scripts by providing a container that receives connections from the Asterisk server, parses the request and calls your scripts mapped to the called URL.

It is actually quote from the original Java API manual. We quote it here just for showing the level of problems application developers will need to deal with during accepting a new API. Actually it is not the Asterisk only problem.

So our final problem is: yes, we have got an API that let us implement the basic things, but the cost of that (not the software performance but a cost for the development) is too high. Especially if we will keep in mind that any hardware related integration will require a constant redevelopment/redesign for the software part.

The way we've decided to go is a new layer in the software development architecture. We've suggested a new layer for Asterisk development. But instead of introducing our own API we've decided to go with HTTP.

In other words we've created HTTP gateway, that lets developers talk with Asterisk via HTTP requests/responses. And Asterisk related applications (scripts at the original) becomes in this model just ordinary CGI scripts.

All the complexity for the above mentioned API's, protocols, managers etc. is hided within our gateway. Application developer now is an ordinary web developer. He/she can simply see a list parameters for incoming request (parameters in HTTP request) as well as the format (list of the parameters) for the response.

What kind of requests our gateway should support? For this let us go back to our introduction. There is a list from the 5 basic services. They are basic things for our gateway too.

OUR IMPLEMENTATION

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 - CallerIdName, caller's number - CallerIdNumber 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.

Originally this approach (map telecom events to HTTP) was developed by D. Namiot (AbavaNet) and used to map INAP and CSTA stuff into CGI requests.

A.Ustinov has done a first implementation for Astersik as his master thesis in Moscow State University. On practice this approach was used by Abava - telecom services for smart house (M.Sneps, Latvia). Service receives a call, reads data from water meter (M-Bus) and passes it via text to speech service back to the user.

The following picture illustrates the common schema:

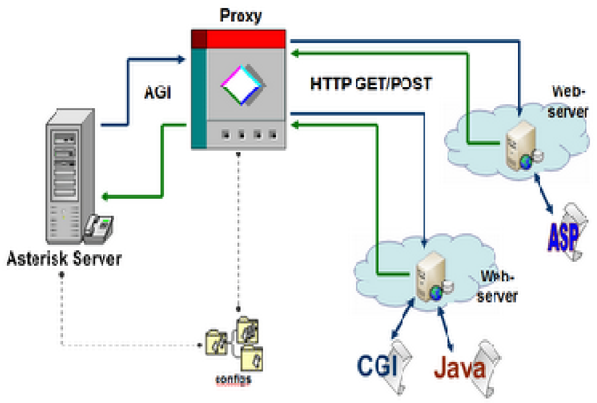

Figure 1
The schema

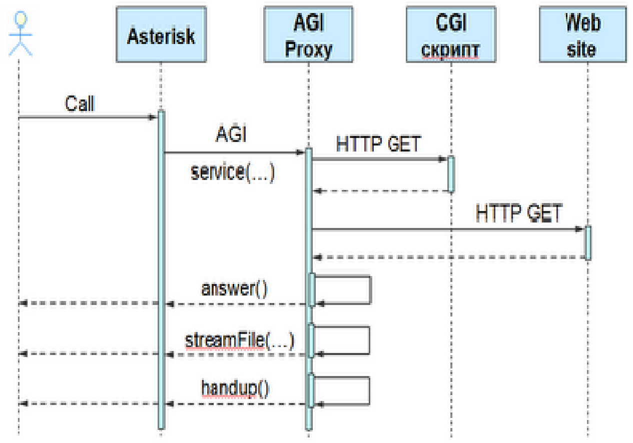

Figure 2
Call flow

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 messages 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 etc.

Under this architecture the sequence of calls is reduced to a very simple diagram.

Source code and files necessary to install the demonstration examples are available here: http://code.google.com/p/asterisk-web-gate/.

CONCLUSION

With this tool we've replaced the traditional telecom programming with web programming. And we can list here at least the following results:
- the entry barrier for programmers of new telecommunication services becomes much more lower. Obviously, there are more web programmers than telecom programmers
- now we have the possibility to develop services using various technologies, such as CGI, JSP, ASP.NET.
- it is possible to integrate new external services without any changes on the PBX side.

References

[1] http://www.asterisk.org/ - Asterisk

[2] http://asterisk-java.org/- Java Asterisk API

[3] http://asterisk.linkstore.ru Web gate for Asterisk