VOIPINTEGRATION
VoIP Control and Processing System

F. J. Serrano, G. G. Talaván, B. Curto, V. Moreno, J. F. Rodríguez Aragón and A. Moreno
Faculty of Sciences, University of Salamanca, Salamanca, Spain

Keywords: Technology integration, VoIP, Telephony.

Abstract: Analog telephone networks, are gradually giving way in favor of new Internet-based communication systems (VoIP). However, the diversity and heterogeneity of systems and suppliers prevent users from communicate with their contacts on an uniform and simple way. In this paper a new VoIP multiprotocol and multidevice system, extensible by plugins, is presented. Heterogeneous character has lead us to the need of considering several different communications mechanisms at present. The usage of common devices for telephone communications has also been considered. A complete system has been developed in order to integrate all possible components of a VoIP system.

1 INTRODUCTION

Analog telephone networks, are gradually giving way in favor of new Internet-based technologies. There are more and more protocols like SIP (Rosenberg et al., 2002) or IAX (Spencer et al., 2009) and systems that use the Internet as communication channel and possible replacement for the analog telephone network. Skype (Skype, 2009) with its own protocol seems to have a dominant position in this area, although many providers already offer free calls from computer to computer, or cheap calls to landlines and mobiles phones mainly using the SIP protocol.

One of the main barriers preventing the proliferation of these new technologies is just the diversity and heterogeneity of systems and suppliers. If end users want to take full advantage of these technologies, difficulties arise like the installation of different clients, learning their particularities, the lack of organization, and so on. Although these clients are very easy to use, just the simple need to interact with a computer may lead some users to prefer to keep using their landline phones as usual.

Up to date, Asterisk (Asterisk, 2009) was the only system that integrates several VoIP technologies. However, Asterisk is server-oriented, i.e. it's not oriented to final users. A platform bringing together the different technologies and allowing users to access them in a uniform way and, if possible, be adapted to the habits acquired over the years by telephony users

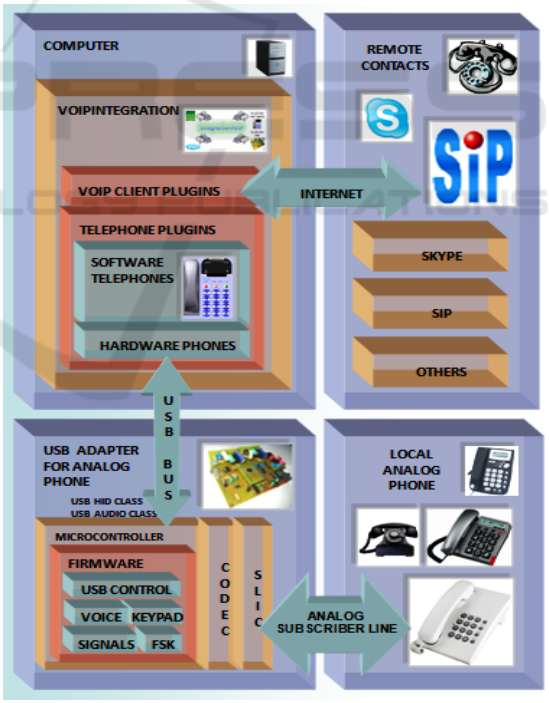

Figure 1: Proposed system.

becomes necessary. The development of an integration platform is viable since specifications of VoIP protocols and clients are well defined, and also those of the subscriber loop and analog phones. The work in this paper involves the development of a plugin

Serrano F., TalavÃąn G., Curto B., Moreno V., RodrÃ■guez AragÃşn J. and Moreno A.
VOIPINTEGRATION - VoIP Control and Processing System.
DOI: 10.5220/0002809301950198
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

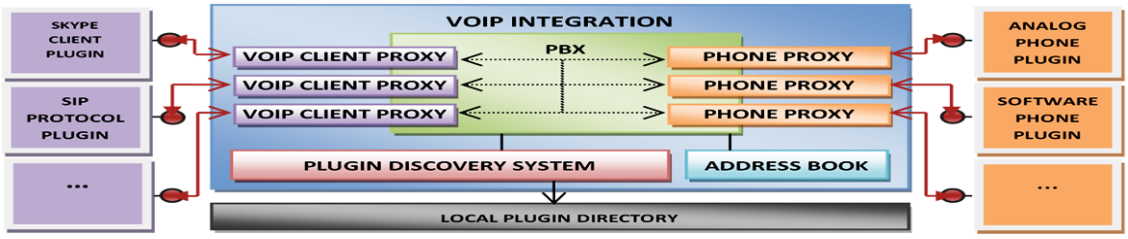

Figure 2: VoIPIntegration architecture.

based communication platform (VoIPIntegration) that is expected to facilitate access by a large number of users to VoIP technologies, to allow users to access all services in the same way that they access the conventional analog phone, and to provide advanced features for more experienced users. Below, we will explain this communication platform we have designed an implemented, we will show how new plugins are integrated in the system to provide compatibility with telephone devices and VoIP protocols, and finally we will describe the already developed plugins.

2 PROPOSED SYSTEM

The system kernel is a application called VoIPIntegration that must be installed on the user's computer. This application lets integrate different telephone devices like hardware and software telephones, coordinate them and easily access IP telephony using popular clients or most relevant protocols (fig. 1).

Platform design is plugin based. On one side, plugins for connection with VoIP clients, like Skype or SIP protocol, are found. These plugins allow every telephone devices connected to the system to access the Internet and to contact other clients using their respective protocols. On the other side, plugins for connection with local telephones (software and analog telephones) are found. To achieve integration with analog phones, we have used the USB port and a hardware adapter that we have designed and developed.

3 VOIPINTEGRATION ARCHITECTURE

The platform kernel (fig. 2) performs the typical functions of a private branch exchange (PBX) and an address book. It establishes connections between telephone devices and VoIP clients with device occupation management and configurable relationships between VoIP accounts and telephone devices, so that calls are received by different phones depending on their origin account. This becomes even more useful by allowing users to maintain multiple active calls simultaneously. As shown in figure 2 one of the software components works as a switchboard. This PBX connects telephone devices installed in the system among themselves or with VoIP clients through plugins. Access to each plugin is done through proxies. Each proxy accesses a plugin through a well-known interface and using a specific protocol. Proxies are responsible for getting the required information from the plugins, storing their state and providing as much functionality as possible in order to make the implementation of plugins as simple as possible.

Specifically, a phone plugin proxy stores the phone status (on-hook, off-hook, dialing, ringing, talking), the name of the audio input/output devices that the phone uses, the default VoIP account used by the phone (although any account integrated in the system may be used), and the establishment of calls by the three available dialing modes (normal, letters and T9). Several instances of a same telephone plugin can coexist.

VoIP client proxies link the PBX with client plugins. These plugins can't have several instances, but they can access several accounts so that each one is seen by the PBX as a different communication channel.

4 VOIPINTEGRATION API

VoIPIntegration has its own API that specifies how different plugins must connect and communicate with the system. This API is based on a generic interface for each plugin type (telephone or VoIP client) and on the format of the text messages exchanged between the application and different plugins. This API also allows extending and supporting new protocols and new phone devices. For this, plugins must fulfill certain requirements and respect the VoIP integration communication protocol. Plugins are not able to start a communication by themselves, but they must wait a VoIPIntegration request. With this plugin design, it's not necessary to use any library, call any function or

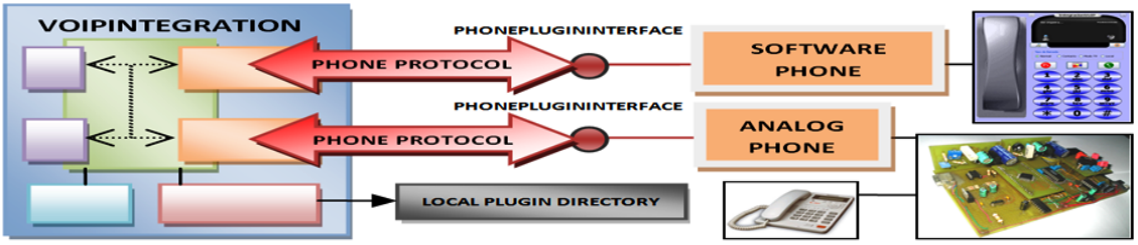

Figure 3: Phone plugins integration.

know anything about VoIPIntegration. This implies a low coupling design.

4.1 Phone Plugins

Communication between VoIPIntegration and phone plugins (fig. 3) is performed by an interface (*PhonePluginInterface*) and using an specific protocol (Phone Protocol) that we have defined. Basic services must be provided by the plugin implementing the interface. Communication protocol messages are used for notifications, for advanced features and for adding new functionality in the future keeping backwards compatibility.

Basic services that the plugin must provide are: to communicate with the user in order to let him know about the reception of calls and to provide him some kind of interface so the user can also make calls; to know at every moment the status of the phone (on-hook, off-hook) and to associate with an input and an output audio device of the system.

Protocol between VoIPIntegration and a phone plugin (Phone Protocol) allows bidirectional communications between them by a plain text messages exchange. Message structure contains a command field and some attributes. Plugins send requests to VoIPIntegration using the return value of an PhonePluginInterface function (*getEvent*). This function will be called continuously by VoIPIntegration. To avoid busy waiting, plugins must internally block this function until they need to send any request. If a plugin doesn't fulfill this requirement, or send incorrect messages, VoIPIntegration would deactivate it. VoIPIntegration responds all the request made by plugins calling another PhonePluginInterface function (*eventResponse*). Response may be the *"OK"* string or the *"ERROR"* string followed by a textual description of the error that happened. Several commands can be sent by means of a numeric sequence to provide advanced functionalities of VoIPIntegration (address book, dialing methods, voice synthesis, etc.) in phone devices without graphical interface. VoIPIntegration can also send orders and notifications to the phone plugins through PhonePluginInterface.

Extra features from the plugins are allowed by combining the *"ACTION"* command from the *Phone Protocol* with the *getActions* function from the *PhonePluginInterface* interface. Plugins must send to VoIPIntegration the name of additional features that they provide using the return value of the *getActions* function. These names are included in the VoIPIntegration graphical interface and when a user selects any of them, an *"ACTION"* command is sent to the corresponding plugin with the name of the special action that must be performed.

Already existing methods are used to extend the protocol with new commands. Thus, plugins compatibility with new versions of VoIPIntegration are ensured. New requests between VoIPIntegration and plugins will be performed in the common way.

4.2 VoIP Client Plugins

Communication between VoIPIntegration and VoIP plugins is performed through an interface (*VoIPClientPluginInterface*) and with a specific protocol (fig. 4) in a similar way as phone plugins described in the above section. VoIP client plugins must provide access to at least one IP telephone account that allows starting calls, receiving them and hanging them up. Plugins must also play and extract audio from devices indicated by VoIPIntegration. For this, an audio library is provided with the application. Communication between VoIPIntegration and VoIP client plugins is bidirectional and uses the same mechanisms as telephone plugins (*getEvent*, *eventResponse*, *getActions*, etc.) although the exchanged messages are different.

5 DEVELOPED PLUGINS

VoIPIntegration provides a GUI in order to manage and control the plugins and the address book. It also provides a library to manage the audio devices of the system that allows recording, playing and synthesizing voice from text. VoIPIntegration includes a plugin discovery system that allows automatically recogniz-

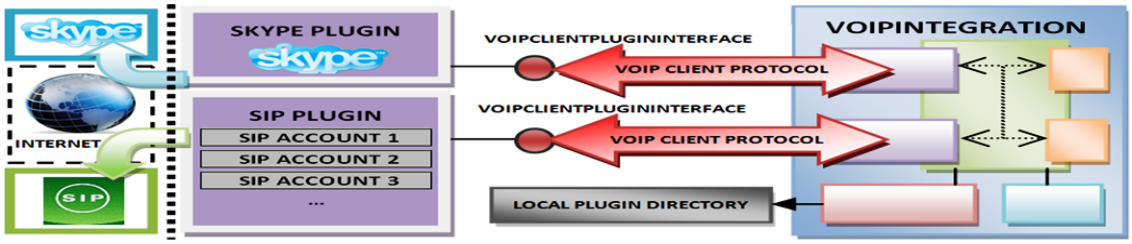

Figure 4: VoIP clients integration.

ing them. For implementing it, Java facilitates that work as it allows us to pack plugins into a jar files with information about them in the manifest.mf file. We will now describe the VoIP clients and telephones developed plugins (Skype, SIP, software telephone and USB telephone).

Skype plugin allows sending and receiving calls through it, sending DTMF tones and additionally importing Skype address book. Skype must be installed in the system and logged in for the plugin to work.

SIP protocol plugin uses open source library MjSIP (Veltri, 2005) in order to send and receive calls. It lets VoIPIntegration the use of several IP telephone accounts with any provider using this protocol. All SIP accounts are managed concurrently, so that several incoming and outgoing call can coexist simultaneously through different accounts.

Software telephone plugin implements a virtual telephone with a friendly graphical interface for VoIPIntegration that uses all the API functionality.

USB adapter plugin allows VoIPIntegration to access to the analog telephone adapter that has been built. Plugin can connect to the phone, retrieve its status, send and receive DTMF tones, send and receive ringtones and send caller ID information. It allows any analog phone connected to the adapter to send and receive calls using any of the VoIP clients integrated in VoIPIntegration. The developed USB adapter fulfill the *USB Audio Class* and the *USB HID Class* standards. Therefore, the prototype doesn't need specific drivers for Windows and Linux, but generic drivers provided by the operating system that are automatically loaded.

6 CONCLUSIONS

A voice communication system in order to work in an heterogeneous VoIP environment has been developed. Main VoIP systems and classic telephone devices have been considered in their specifications. System design has been made in a way that, in the future, new services of this technological scope can be incorporated. For this, we have designed a software that can handle plugins. Different plugins for the most popular technologies like Skype client or SIP protocol have been developed in order to validate the proposed design.

From the users point of view, different interfaces are also supported in our plugin-based proposal. So, a software telephone and its corresponding plugin have been developed. An USB adapter, which allows interaction between the system and analog phones, has been developed and built in order to achieve a greater usability. It's been necessary to design and develop a complete hardware system with its corresponding embedded real-time control software. The prototype has been tested with different analog telephones and satisfactory results have been obtained.

This proposal allows accessing different VoIP technologies in a simple way at low cost. So, users may integrate their telephone systems with current VoIP systems performing a small investment using our proposed architecture.

ACKNOWLEDGEMENTS

This work has in part been financed by Junta de Castilla y León (Project SA030A-07) and Spanish Ministry of Science and Innovation (DPI2007-62267). The main author has also worked under the support of an University of Salamanca fellowship.

REFERENCES

Asterisk (2009). Asterisk reference information version 1.6.1.6. Asterisk.org.

Rosenberg, J. et al. (2002). Sip: Session initiation protocol (rfc3261). Network Working Group.

Skype (2009). Skype public api 4.1 reference guide. In *Skype API reference*. Skype.

Spencer, M. et al. (2009). Iax: Inter-asterisk exchange version 2 (rfc5456). Network Working Group.

Veltri, L. (2005). Mjsip version: 1.5. In *MjSip-Mini-Tutorial*. MjSIP.