A VOIP PLATFORM AS A VIRTUAL PBX SERVICE

Walter Balzano, Maria Rosaria Del Sorbo and Mario Epifania
Department of Physical Sciences, University of Naples "Federico II", Via Cinthia, 4, 80126 Naples, Italy

Keywords: VoIP, IP-PBX, systems vitualization, Asterisk, Isolation, easy system restore.

Abstract: Voice over IP telephony, allowed by large band and low cost communication through Internet, is the base of a project of a communication platform for IP-PBX. The new idea is to use open source software to implement an IP phone switch system by a business LAN, where a virtual PBX can provide services and reliable communication. The challenge is to virtualize many systems in one physical system using Linux vservers resources. This approach yields many advantages: easy restore from crash events, security and simple resource management. The drawback is a heavier hardware load, solvable using more powerful CPU and memory architectures.

1 INTRODUCTION

Latest communication achievements, such as large-band transmission technologies and audio-video more efficient compression algorithms, have allowed recent speed growth in the Web data flows. These innovations have generated protocols with faster multimedia content transmissions and shorter response times (Mauro, 2005). The VoIP, Voice over Internet Protocol (Davidson, 2006), is a part of these new technologies: it aims to phone communicating by the World Wide network (Wallingford, 2005).

One of the advantages of VoIP communicating systems is a real reduction of phone services costs, due to a good trade-off between geographic distances and bandwidth employment allowed by package commutation (Clark, 2001). Moreover, VoIP makes possible new services creation, based on the integration of audio, video and other information data on the same multi-service network (Chatterjee, 2005). The integration of traditional phone services with data exchange on a IP business network in one communication infrastructure, is called IP-PBX. In fact, IP-PBX is a software running on a server to control all connection and composition procedures of internal and external Local Area Network communications. IP-PBX doesn't need any special hardware (Ohrtman, 2003): traditional PBX (Private Branch eXchange) functions are managed associating phone numbers to terminal IP addresses, properly coordinating the connection requests (Sulkin, 2002).

IP-PBX systems connect many communication media, providing new services such as digital receptionist, IVR tree, voicemail, business databases access by the Caller ID and so on.

This project presents a Small Medium Business (SMB) VoIP solution: an Open Communication Platform to allow IP phone switch system implementation. Costs optimization has been gained using open source software and a specific VoIP, Session Initiation Protocol (SIP).

2 THE COMMUNICATION PLATFORM

2.1 Virtual IP-PBX

Currently, IP-PBX is based on three standards:

- **IP-enabled PBX** (TDM is the main technology but IP cards can enable VoIP on some emplacements).
- **Hybrid IP-PBX** (integration of packet and circuit commutations with advantages from both approaches but with the double wiring drawback).
- **Full IP-PBX** (based on devices satisfying QoS constraints (Kist, 2003), with efficient security functions).

Balzano W., Rosaria Del Sorbo M. and Epifania M. (2007).
A VOIP PLATFORM AS A VIRTUAL PBX SERVICE.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - SAIC*, pages 303-308
DOI: 10.5220/0002409903030308

A **Virtual** (or **Hosted**) IP-PBX system implements its own functions as a service without physical PBX devices, reducing the costs; all the IP-PBX services are provided by a centralized infrastructure. Open source software allows further economy, even if it needs upgrades and bugs monitoring. Asterisk, an open source Digium's PBX platform, provides PBX services. It is designed for maximum flexibility by Application Programming Interfaces (API), defined around a central PBX core system. This advanced core handles PBX internal interconnection, plain abstracted from specific protocols, codecs, and hardware interfaces from the telephony applications. This feature makes Asterisk independent by specific hardware.

2.2 The Architecture

Our research is focused on the definition of an Open Communication Platform correctly and efficiently managing calls routing.

A software modular architecture supports this implementation and is based on SIP Entities. SIP is remarkable for its "lightness" and compatibility with other IP protocol and open source software (Abbasi, 2005). SIP entities follow the client/server model and interact by messages exchanges and error code feedbacks as shown in Figure 1.

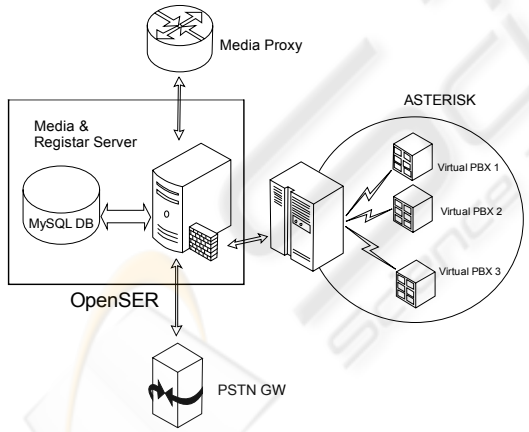

Figure 1: Platform Architecture scheme.

Main architectural components are:

(**A) PSTN Gateway (PSTN GW)**, representing a bridge between IP network and the traditional telephony, using a LCR (Least Cost Routing) policy, which exploits cheaper outbound providers (Yuan Zhang, 2002). This module has been implemented by an Asterisk instance on a server, sending outbound calls on PSTN by trunks, connection channels configured by the dialplan (Van Meggelen, 2005).

(B) Proxy and Registrar Server, routing call flows and registering IP-PBX user accounts. This device works like a *server* for the call originating machine and like a *client* for the destination terminal. So, the call management is delegable, limiting resources requirements on user terminals.

The proxy and registrar functions are implemented by OpenSER (Open SIP Express Router), an open source SIP server. It is written in pure Unix/Linux system C language, with specific architecture optimizations to offer high performance and works as net speed balancer.

This module independently performs user account registrations on a Mysql database, supporting Asterisk PBXs and limiting their work to PBX functions: this can result in more scalable architectures.

Proxy server operation is tied to the Media proxy features and activity (Stojsic, 2001).

(C) Media proxy, a solution to the Network Address Translation (NAT) Traversal problem, affecting SIP protocols (Tam, 2002). These protocols suffer NAT environments, because during call signalling phase, the caller send to the called the address to re-contact it on the net; but the called regards its LAN address as its absolute address and, when the called tries to contact it on the net, it can't route the packets.

The Media Proxy acts as a mediator in this troublesome routing.

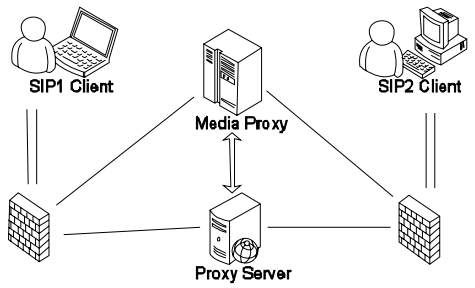

Figure 2: Media Proxy at work.

Figure 2 shows the Media proxy operation: a correct procedure is guaranteed by a Proxy Server, which establishes the SIP between two user agents.

The main Media proxy features are:

- to ensure SIP client transparent operation behind NAT nets,
- to manage all multimedia session flows,

- to distribute RTP traffic load on multiple proxy servers, running on multiple hosts,
- to specify Proxy Server use by the caller/called SIP domain.

(D) IP-PBX SWITCH, representing the platform core. IP-PBX SWITCH consists of all Virtual Subnet IP-PBX, managed by the service supply company. Every Virtual IP-PBX instance is implemented by Asterisk.

Investigating on this main module we conceived an innovative and efficient virtual architecture, shown in next paragraphs.

Finally, the entire platform is administrated by multi-layer interfaces, which monitors every module and provide IP-PBX services.

2.3 Virtual Asterisk PBX

Every service supply subnet needs its own PBX, even if virtualized and remote accessible. Then, a double problem rises:

- A hardware cost problem, to buy a dedicated Asterisk server for each subnet.
- A space problem, because servers need room and many servers can crowd the business server farm.

This situation isn't acceptable for SMB reality, because hardware cost and space problems depress profit strategies and limit business increase.

The virtualization strategy involves an accurate study of all steps from the basic concept of virtual machines to the most important virtualization techniques:

- emulation: a full system simulation, or "full virtualization with dynamic recompilation" where the virtual machine simulates the complete hardware, allowing an unmodified OS to run in a really different environment.
- paravirtualization: the virtual machine does not simulate hardware but offers a special API that requires OS modifications.
- native virtualization and "full virtualization": virtual machine only partially simulates hardware, allowing an unmodified OS to be run alone, but the guest OS must be designed for the same type of CPU. "Native virtualization" also stands for hardware assistance through Virtualization Technology.

Next step provides an IP-PBX switch system model. Asterisk machines, in fact, can be virtual machines and they can remarkably limit the space occupation of real Private Client Services (PCS). However this solution might be insufficient to reduce high hardware cost: *a*) managing several virtual machines on one real machine requires a powerful computer; *b*) virtual machines are a kind of heavy virtualization that steps down to the physical layer and needs a partition to be replicated on every virtual machine.

A virtual server is shared by more users; each of them can administrate his space and his resources as a single dedicated server.

To understand how they work and what kind of different virtual machine they support, it's necessary an accurate comparative analysis of hypervisors, or Virtual machine monitors, such as vmware workstations, Qemu, Xen and Virtuozzo.

The open source **Linux-Vserver project** is used to implement our virtual server system.

Vservers, as Xen hypervisor, are a different approach to Virtual Machine Monitor: Xen uses only one kernel for each virtual server, it has minimum overhead and allows many UNIX tasks.

This solution is based on user environment space division in different Virtual Private Servers (VPS) entities: each VPS appears as a real server to locally running processes.

Different Linux distributions use patched kernel to supply special support to unusual hardware or to extra features: Linux Vservers allow different Linux distributions to run simultaneously on a single patched shared kernel, without direct access to hardware and sharing resources in an efficient way.

The kernel main purpose is to build a true and own abstraction layer on the hardware to allow processes (tasks) to work and operate on resources (data) without knowing underlying hardware details.

Ideally this processes would be completely hardware unaware, written in an interpreted language and also don't require any hardware specific knowledge.

More processes can run simultaneously, so the kernel has to assign a time-machine slice and hardware access to each task (multitasking). Each virtual server has got its own packets, services, users and it's limited to use only some IP address and only some file system zones.

We can think a virtual server as a new system using chroot system call concept with specific root user and its own configurable resources manager, different hostname and IP address.

Our idea was to use an Asterisk vserver for each IP subnet requiring PBX services, to exploit the simple management of its features, as voicemail, to integrate them with the software functions.

Asterisk Vservers represent a virtual PBX instance and more vservers can run on a single physical machine (Des Ligneris, 2005). Each virtual system created by Linux vserver has its own root access and every owner (subnet using the service) can load and execute Asterisk PBX features: this is an example of a so called Virtual Private Asterisk (VPA).

Therefore, the vservers hosting device holds in a relatively small box an efficient phone exchange switch and the patched kernel is shared between VPS with proper strategies. This technology introduces many profits: first, every VPS can work at the maximum performance allowed by the system. In fact, processes within virtual server run as regular processes on the host system. Then, it is possible to develop an action more memory-efficient and I/O-efficient than the whole-system emulation, which cannot return unused memory or share a disk cache with the host. Furthermore, VPS overhead is pointless, because there is a shared kernel for all VPS and no hardware emulation.

Regarding security concerns, every VPS is contained into its relative security context: a VPS can't directly access the host system kernel without licence and there are several security layers. For example, other VPS users and owners cannot load our modules or alter kernel directly if they aren't root. Above discussion points out the opportunity of huge cost reductions for SMB that want to invest in the new VoIP PBX market.

2.4 Isolation Model

Isolation between Vservers simultaneously running on the host is guaranteed by kernel modifications:

- Context Separation
- Network Separation
- The Chroot Barrier
- Upper Bound for Caps
- File system XID Tagging

Context separation consists in hiding to a given process all the processes out of it and in forbidding every interaction between processes belonging to different contexts. It needs some data structures extensions to make processes context sensitive and to allow differentiation of identical User ID (UID) exploited in different contexts.

There is a default context, oriented to guarantee the host boot and a special context, called Spectator, that ensures a global host processes sight. Network separation limits processes into an IP address space available for the host.

Some isolation problems can rise: a process might perform binding operation on a special address, IPADDR_ANY for example, and this can be done without modifying any process running in the vserver. To reduce total system overhead we don't use virtual network devices, acting on kernel socket binding system for packet transmission.

For our project we configure the Vservers subnet as a C class subnet (192.168.100.xxx). Host machine containing vservers has the fixed address 192.168.100.253; other IP addresses about Asterisk instances vservers have progressive numbers starting from 1 (for instance VPBX1 has IP 192.168.100.1, VPBX2 has IP 192.168.100.2 and so on). Figure 3 shows the network address scheme:

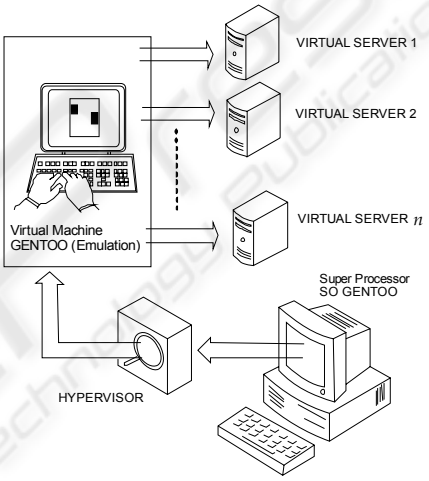

Figure 3: Complete virtual subnet.

An example of a code segment follows (shell command lines for the virtual subnet creation):

```
>> host# vserver-new pstn-gateway --hostname
PSTN-GW --context 1249 –interface
eth0:192.168.100.249/24 template ./gentoo-
template-x86-20060317.tar.bz2 x86

>> host# vserver-new H323-asterisk --
hostname H.323 --context 1250 --interface
eth0:192.168.100.250/24 template ./gentoo-
template-x86-20060318.tar.bz2 x86

>> host# vserver-new MySQL --hostname MySQL
--context 1252 --interface
eth0:192.168.100.252/24 template ./gentoo-
template-x86-20060319.tar.bz2 x86

>> host# vserver-new VPbx01 --hostname
VPbx01 --context 1001 --interface
eth0:192.168.100.1/24 template ./gentoo-
template-x86-20060320.tar.bz2 x86
```

When chroot(), open() or fchdir() system calls are executed, in the next call there will be an

information loss; Linux Vserver manages this event by a simple and efficient method: a mark, called Chroot Barrier, placed in vserver father directory which disallows going out of boundaries.

Linux doesn't implement the file system POSIX features that would make safer the setuid and setgid executables: it's safer to set an upper bound for all context processes adding a supplementary capability mask which limits all processes to belong to the mask context.

File system eXchange station ID (XID) tagging increases context isolation and enable Context Disk Limit and Per Context Quota Support on a shared partition between more Vservers. It's difficult to manage the context ID entity for every file, because it needs file system representation disk modifications or the extraction of some bits from existing data structures. It's possible the storage of XID, for instance, implementing a not invasive solution, that is to use the most significant bit of UID and GID. Now we evaluate the security profits presented by vservers.

A corrupted vserver can't affect and damage the host server. The host can be used to study in a safe manner the crashed vserver with the purpose to repair it: this is a very innovative feature that can be performed by no other Linux installation. Vservers mobility is another good characteristic of our IP-PBX system: after mainframe system crashes it's very easy to move vservers from a machine to another (high disaster recovery). Vservers, IP-PBX system instances, are just simple directories in the host machine and then full portable: we can quickly restart the services stopped in the crashed machine.

So, our IP phone exchange switch system contains all Asterisk PBX instances (representing every business virtual PBX) on a single company Mainframe running a Gentoo-distro: Linux Gentoo offers a compiling system better than the traditional pre-compiled binaries and permits flexibility and performance optimization. Users, through Gentoo Portage Settings, can easily customize every system packet, compile the Kernel, as the system requires, and produce executables fit to their own platform.

3 CASE STUDY

Virtualization is a part of the entire project, starting from the PBX system architecture and arriving to its implementation. The physic box is the mainframe placed into business company which contains the Asterisk vservers instances; it must have high CPU performances and large memory space to provide an optimal quality of VoIP services.

The model using a single machine for many Asterisk vservers can be helpful but, at the same time, hazardous for system reliability: host machine hardware crash events can corrupt the whole service and require long times to repair, wasting performance indicators. The introduction of machine virtualization concepts can help us in solving this problem, replicating the host Gentoo machine by a virtual machine.

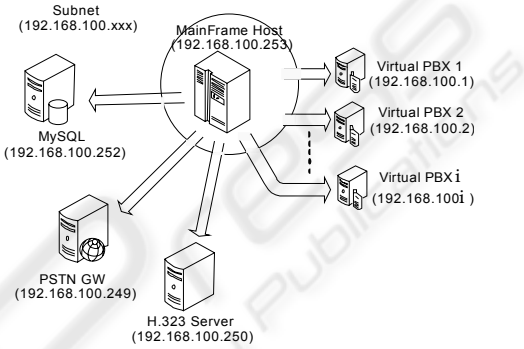

Figure 4: Virtual machine emulation.

This solution has remarkably increased the degree of reliability of the system, because the system crashes can be easily resolved: in case of breakdowns it's possible to restore the system in a short time using a backup copy of the virtual machine. Moreover, it's easy to move the virtual machine for system platform upgrading, without constraints in hardware reconfiguration in another host. The framework of a virtual machine is saved as single row and so it's easy to backup, to copy and to move it. The double encapsulation model is related to a double layer isolation: the virtual telephones exchanges (the Vservers) are conceived to be executed on a virtual machine, contained in a physical machine. Then, there is a really "protecting" shell that closes the physical machine and looks after it from problems and errors.

Thus the machine stays lastingly undamaged and could be hardly corrupted. Therefore, even if the virtual machine is corrupted and damaged, the physical machine isn't ever involved and then the virtual machine can be quickly resumed using its backup copy. There is a drawback behind the advantages to have this double virtual layer with high fault tolerance and system scalability: a heavier load for the mainframe, forced to support the elaboration of a light emulation (vservers) in one heavy emulation (virtual machine). Nevertheless it can be regarded as a slight detail if we consider the

advantages introduced by virtualization, because the provider subnet mainframe must have such computing characteristics that it can guarantee a fast process of all requests from an IP-PBX, and this indicator influences VoIP QoS through the maximum efficiency provided (Hardy, 2003).

4 CONCLUSIONS

This project is a new VoIP technologies application which implements services management methods of VoIP PBX systems by virtualization techniques.

Platform architecture has been built step by step tracing a safe and scalable model, and this technology can be considered as a starting point for future researchers, who want to follow a development of the virtual machine potential applications to obtain several profits in data processing.

With the aim of monitoring our virtual switch system we have used Cacti, a "complete network graphing solution designed to harness the power of RRDTool's (Round Robin Database Tool) data storage and graphing functionality. It provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrapped in an intuitive, easy to use interface that makes sense for LAN-sized installations up to complex networks with hundreds of devices".

However, virtual servers opens new horizons in systems management.

With few efforts we can realize on a single physical system, with the required hardware features, many absolutely independent and cooperating systems.

So, from the central system we can administrate and monitor all these virtual systems, simplifying the management operations and the eventual recover or backup actions.

The security field is the most important future development of virtual server: by intrusion detection tools, we can easily realize a safe host isolated by the system, for specific use, fully manageble from host system. Then in practise we can "inbox" our firewall and fight hackers that often try to access our systems.

Finally, the testing environment can consider virtual servers as separate systems, where it's possible to execute various tests and experiments which won't affect the main host system.

REFERENCES

Abbasi, T.et al., 2005, A comparative study of the SIP and IAX VoIP protocols, *Canadian Conference on Electrical and Computer Engineering*, pp. 179-183.

Chatterjee, S. et al., 2005, SIP-based enterprise converged networks for voice/video-over-IP: implementation and evaluation of components, *IEEE Journal on Selected Areas in Communications, Vol. 23, Issue 10*, pp.1921 – 1933.

Clark D. D. et al., 2001, *Internet Telephony* , MIT Press, Boston MA.

Davidson, J. et al., 2006, Voice over IP Fundamentals, *Cisco Press, Indianapolis, IN*.

Des Ligneris, B., 2005, Virtualization of Linux based computers: the Linux-VServer project, *HPCS 2005, 19th International Symposium on High Performance Computing Systems and Applications*, pp.340-346.

Hardy, W. C., 2003, *VoIP service Quality*, McGraw-Hill Professional, New York, NY.

Kist, A.A.; Harris, R.J., 2003, Using virtual SIP links to enable QoS for signaling**,** *ICON2003, The 11th IEEE International Conference on Networks*, pp. 301 – 306.

Mauro D.R., Schmidt K.J., 2005, *Essential SNMP*, O'Reilly, Sebastopol, CA.

Ohrtman, F., 2003, *Softswitch Architecture for VoIP*, McGraw-Hill Professional, New York, NY.

Stojsic, G.et al., 2001, Formal definition of SIP proxy behavior, *EUROCON'2001, International Conference on Trends in Communications,* Vol. 2, pp. 289-292.

Sulkin A., 2002, *PBX Systems for IP Telephony*, McGraw-Hill Professional, New York, NY.

Tam, K.K.; Goh, H.L., 2002, Session Initiation Protocol, *IEEE ICIT '02, IEEE International Conference on Industrial Technology,* Vol. 2, pp.1310 – 1314.

Van Meggelen J. et al., 2005, *Asterisk: The Future of Telephony*, O'Reilly, Sebastopol, CA.

Wallingford T., 2005, *Switching to VoIP*, O'Reilly, Sebastopol, CA.

Yuan Zhang, 2002, SIP-based VoIP network and its interworking with the PSTN**,** *Electronics & Communication Engineering Journal*, Vol. 14, Issue 6, pp.273 – 282.