ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“IMPLEMENTACIÓN DE UN SISTEMA PARA LA GENERACIÓN AUTOMÁTICA DE LLAMADAS SALIENTES EN UN CALLCENTER”

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN TELEMÁTICA

Presentada por:

MELINA MARCELA PONTÓN LOAIZA

WENDY PAOLA YÁNEZ PAZMIÑO

GUAYAQUIL – ECUADOR

AÑO

2011

AGRADECIMIENTO

A Dios, por darme la fuerza para seguir adelante cada día, a pesar de las adversidades. A mi familia y amigos, por brindarme su apoyo incondicional. A mis maestros, por compartir sus conocimientos conmigo.

Melina M. Pontón Loaiza

A Dios, por ser mí luz y guía. A mis padres por ser ejemplo de superación y perseverancia. A mis hermanos, por contagiarme con su alegría. A mis maestros por haberme permitido nutrir de sus conocimientos.

Wendy P. Yánez Pazmiño

DEDICATORIA

A Dios, guía y luz de mi camino. A mis padres y hermanos por ser los pilares fundamentales en mi vida.

Melina M. Pontón Loaiza

A todas las personas que contribuyeron en mi formación profesional, brindándome una sólida formación a través de sus conocimientos y experiencias, permitiéndome culminar una etapa más de mi vida académica

Wendy P. Yánez Pazmiño

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Gabriel Astudillo Brocel

PROFESOR DE LA MATERIA DE GRADUACIÓN

_____

Ing. Patricia Chávez Burbano

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Melina Marcela Pontón Loaiza

Wendy Paola Yánez Pazmiño

RESUMEN

El proyecto realizado consistió en implementar un sistema telefónico basado en Asterisk, para lo cual se desarrolló una aplicación PHP que hace uso de la Interfaz de Administración de Asterisk (AMI), capaz de generar automáticamente llamadas salientes programadas en base al monto adeudado por el cliente. Para esto se estableció como escenario de prueba una empresa de gestión de cobranzas, que mantiene una base de datos de los clientes que poseen una tarjeta de crédito, dicha base es actualizada mensualmente por el administrador del sistema.

Al momento de ejecución de las llamadas se valida que la grabación fue escuchada, si no hubiese sido así se programará la realización de una segunda llamada. Adicionalmente, cada mes se genera un reporte de las llamadas realizadas a los clientes, el cuál será entregado a la empresa.

Con la realización de este proyecto se pretendió administrar de forma adecuada los recursos tanto económicos como tecnológicos de la empresa, ya que se evita el pago de una persona que realice las llamadas y además se hace eficiente la realización de las mismas.

ÍNDICE GENERAL

ABREVIATURAS

AJAM	Administrador de Asterisk con JavaScript Asíncrono
AMI	Interfaz de Administración de Asterisk
ANI	Número de Identificación Automática
BRI	Interfaz de Taza Básica
DAHDI	Interfaz de Dispositivos Digium de Asterisk
DID	Llamada Interna Directa
FXO	Interfaz de Central Externa
FXS	Interfaz de Abonado Externo
GPL	Licencia Pública General
IAX	Inter-Asterisk eXchange Protocol
MGCP	Protocolo de Control de Dispositivos de Entrada
IP	Protocolo de Internet
PCI	Interconección de Componentes Periféricos
POTS	Servicio Telefónico Básico y Antiguo
PRI	Interfaz de Taza Primaria
PSTN	Red de Telefonía Pública Conmutada
RDSI	Red Digital de Servicios Integrados
SCCP	Protocolo de de Control de Cliente Skinny
SIP	Protocolo de Inicio de Sesión
YUM	Actualizador YellowDog Modificado

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las llamadas de Voz sobre IP a través de Internet, o telefonía por Internet, se han convertido en una manera muy popular de ahorro en la infraestructura de telecomunicaciones de diversas empresas y organizaciones, ya que resultan muy baratas y, en muchas ocasiones, incluso gratis al hacer uso de las redes de transporte de datos para la transmisión de voz, lo que ha resultado en que la telefonía tradicional pierda terreno entre aquellos clientes que se adaptan fácilmente a las nuevas tecnologías, pues todo lo que se requiere es una única conexión a un red IP.

Un ejemplo de esto lo podemos apreciar en las principales empresas del sector público y privado tales como: Ministerio de Salud, Ministerio de Cultura, Instituto Nacional de Contratación Pública, CNT, AeroGal, RTS, Banco ProCredit, UNIVISA, entre otros; en las cuáles se encuentran implementado Elastix, que es un software libre de VoIP basado en Asterisk, desarrollado por una empresa ecuatoriana.

Mediante una adecuada configuración de una central telefónica Asterisk y haciendo uso de los módulos DAHDI podremos comunicarnos con clientes que se encuentren conectados a la PSTN. Por otro lado, la generación automática de llamadas permite tener un control más eficiente de los clientes

que fueron contactados, evitando la contratación de una persona para la implementación de éste tipo de soluciones.

CAPÍTULO 1

1. ANÁLISIS CONTEXTUAL

1.1 Antecedentes

Los sistemas de comunicación de voz y en especial la red de telefonía convencional, indudablemente influyen en el desarrollo de cualquier empresa. Hoy en día es necesario incluir en el plan de negocios de la empresa una inversión en la implantación de una infraestructura de telecomunicaciones que permita la comunicación tanto entre los diferentes departamentos de la propia empresa, como la interconexión con el exterior de la misma.

Debido al surgimiento de las primeras centrales telefónicas, más conocidas como PBX (Central Privada de Intercambio de Comunicaciones), las empresas evitan conectar todos sus teléfonos a

la PSTN (Red de Telefonía Pública Conmutada, Public Switch Telephone Network por sus siglas en inglés), consiguiendo una infraestructura local de voz, independiente de cualquier proveedor de telefonía, permitiendo hacer llamadas internas sin costo alguno.

Con la digitalización de la señal de voz, el fuerte crecimiento de las redes IP y la aparición de protocolos de transmisión en tiempo real se ha creado un nuevo entorno para la transmisión de voz sobre IP (VoIP), con lo que no solo se ha conseguido reducir costes, si no que se pueden ofrecer nuevos servicios de valor añadido y funcionar independientemente del dispositivo de acceso utilizado (teléfono convencional, teléfono IP, softphone).

1.2 Justificación

Según información disponible en el portal web de la Superintendencia de Bancos, aproximadamente el 34% de la población ecuatoriana hace uso de tarjetas de crédito para realizar sus compras. El 54,30 % de dichas compras utilizan la modalidad del crédito rotativo [2], razón por la cual las empresas de cobranza se ven en la necesidad de implementar un sistema eficiente que permita recordarles a los usuarios datos importantes de sus pagos como son: monto total del

pago, fecha de vencimiento del mismo, número de cuotas vencidas, entre otros.

Aunque esta tarea podría ser cumplida por una persona, estaría sujeta a posibles errores, los mismos que pueden ser evitados implementando un sistema que genere automáticamente llamadas salientes desde una central telefónica. Sin embargo se requiere de un administrador del sistema, que no sólo verifique el correcto funcionamiento del mismo, sino que además actualice mensualmente la base de datos con la información que debe proporcionar la empresa.

1.3 Descripción del Proyecto

Con la implementación de este sistema de generación automática de llamadas se pretende alcanzar los siguientes objetivos:

1.3.1 Objetivo General

Implementar un sistema de marcación telefónica en Asterisk, que permita generar llamadas salientes automáticas desde una central telefónica, basando dichas llamadas en condiciones previamente establecidas por la organización, a fin de mejorar la calidad y eficiencia del servicio.

1.3.2 Objetivos Específicos

- Realizar las configuraciones de los paquetes necesarios para implementar una base de datos en Asterisk.

- Desarrollar una aplicación PHP que haga uso de la Interfaz de Administración de Asterisk (AMI).

- Crear una base de datos en Asterisk con la información que la organización nos proporcione acerca de los clientes, para posteriormente realizar las llamadas salientes respectivas.

- Definir los criterios a tomarse en cuenta para la generación de las llamadas automáticas a los clientes de la organización.

El proyecto a realizarse consiste en la implementación de una centralita telefónica basada en Asterisk capaz de generar automáticamente llamadas salientes, programadas en base al monto que adeuda el cliente en su tarjeta de crédito.

Para el diseño de ésta solución tecnológica se ha establecido como escenario de prueba una empresa de gestión de cobranzas, que mantiene una base de datos de los clientes de las diferentes tarjetas de crédito, esta base de dato será actualizada mensualmente por el administrador del sistema.

El esquema a utilizar en este proyecto se define en la figura 1.1.

Fig. 1.1 Esquema del proyecto

Cada mes la empresa de cobranza recibe la información actualizada de los clientes y ésta es almacenada en la base de datos. El sistema automáticamente realizará las llamadas

al inicio de cada mes, y en caso de que el usuario no haya contestado, se procederá a realizar una segunda llamada a mediados de mes, para recordarle el valor a cancelar.

El monto total del pago se lo obtendrá sumando los valores de las cuotas vencidas que se disponen en la base de datos, una vez obtenido el total se lo ordenará en forma ascendente y es siguiendo precisamente este orden como se efectuarán las llamadas a los clientes.

Con los datos de cada usuario se generará un archivo de audio, el cuál será escuchado por el cliente. El contenido del archivo de audio tendrá el siguiente formato: "*Estimado <nombre del cliente>, Oroquil le informa que adeuda la cantidad de <monto adeudado> dólares, correspondiente a <número de pagos vencidos> meses. La fecha de pago límite es <fecha de pago>. Presione 3 para volver a escuchar el mensaje o 5 para confirmar que escuchó el mismo y terminar la llamada*".

Las llamadas serán realizadas a los números telefónicos de los clientes que se encuentren registrados en la base de datos. Dado que se tienen máximo tres números telefónicos

guardados por cliente, se procederá a llamar uno a la vez hasta que se logre contactar al cliente.

Cuando el cliente presione el dígito 5, se validará que escucho el mensaje completo, en caso de que marque 3 se le repetirá el mensaje. Se validará así mismo el caso en que se presionen dígitos diferentes al 3 o 5, indicándole al cliente del error y pidiendo ingresar una opción correcta.

Si el cliente no contestó a ninguno de los teléfonos registrados, se lo llamará en una segunda ocasión a fin de darle el mensaje.

Una vez que se han realizados las llamadas a todos los clientes, se generará un reporte de todas las llamadas, tanto contestadas como no contestadas.

1.4 Metodología

Para cumplir nuestros objetivos realizaremos las siguientes tareas:

- Instalación de Asterisk sobre un servidor con sistema operativo Centos 5.4.
- Instalación del servicio de base de datos MySQL.

- Instalación del servicio web Apache.
- Instalación del compilador para el lenguaje de programación PHP 5.
- Instalación de la tarjeta Digium TDM410P PCI con 4 puertos FXO en el servidor.

1.5 Perfil de la Tesis

La presente tesis tiene como objetivo principal generar automáticamente llamadas salientes desde una central telefónica Asterisk hacia clientes conectados a la PSTN.

En el capítulo 2, se abarcan los fundamentos teóricos, que permitirán comprender el desarrollo de las comunicaciones basadas en VoIP, sus características, mecanismos de implementación, aplicaciones y servicios que proporciona actualmente.

En el capítulo 3 se detallarán las especificaciones técnicas del sistema, el análisis, diseño y la implementación del proyecto.

Finalmente en el capítulo 4, se realizarán las pruebas de conexión y el establecimiento de las llamadas, con lo que se espera demostrar el funcionamiento del proyecto.

CAPÍTULO 2

2. ASTERISK, FXO/FXS, DAHDI Y AMI

2.1 Asterisk

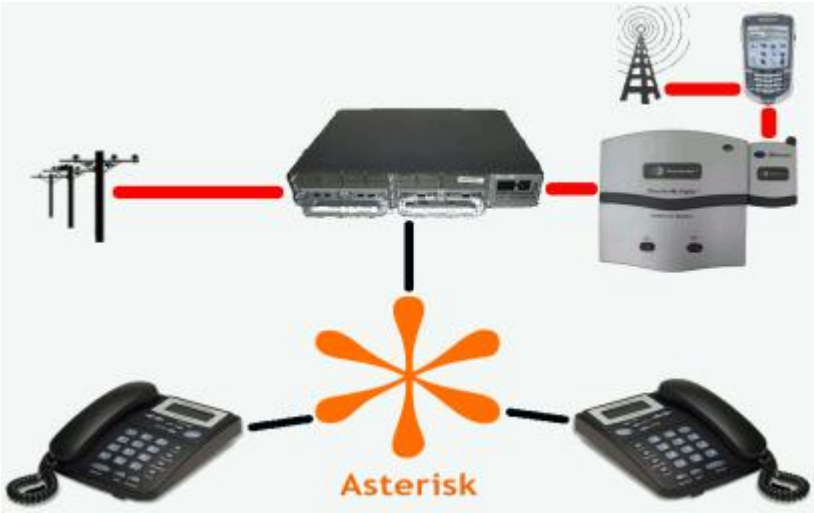

Fig. 2.1 Asterisk, la central telefónica de código abierto

Asterisk – La central telefónica de código abierto, es un software de código libre, bajo licencia GPL, capaz de convertir a una simple

computadora en una central de telefonía IP [3]. Este software proporciona servicios VoIP y puede interoperar con equipos de telefonía estándar básicos usando un hardware relativamente de bajo coste, tal como se muestra en la figura 2.1.

Mark Spencer fue el creador de Asterisk y actualmente es su principal desarrollador, junto con otros programadores que han contribuido a corregir errores y añadir novedades y funcionalidades.

Originalmente, Asterisk fue desarrollado para el sistema operativo GNU/Linux, aunque en la actualidad también se distribuye en versiones para los sistemas operativos BSD (Distribución de Software de Berkeley), MacOSX, Solaris y Microsoft Windows.

Asterisk incluye muchas características que antes de su creación sólo estaban disponibles en costosos sistemas propietarios PBX, como es el caso del buzón de voz, conferencias, IVR (Respuesta Interativa de Vos, Interactive Voice Response por sus siglas en inglés), distribución automática de llamadas, y muchas otras funcionalidades que pueden ser creadas por los usuarios, diseñando un plan de marcado que puede interactuar con scripts desarrollados en lenguajes como: Pascal, Ruby, PHP, Java, Perl, entre otros.

Esta centralita permite la conectividad entre las redes PSTN y las redes IP en tiempo real, para lo cual utiliza las tarjetas electrónicas FXS/FXO fabricadas por Digium u otros proveedores.

SIP, H.323, IAX y MGCP son algunos de los muchos protocolos VoIP que soporta Asterisk. Los protocolos básicamente son utilizados para registrar, autenticar y negociar direcciones IP y puertos, además de controlar el estado de las llamadas.

2.1.1 Funcionalidades de Asterisk

Asterisk incorpora todas las funcionalidades de una centralita tradicional, y muchas más:

- Conexión con líneas de telefonía tradicional, mediante interfaces tipo analógico (FXO) para líneas de teléfono fijo o bien móvil y RDSI (Interfaz de taza básica BRI, o Interfaz de taza primaria PRI).
- Soporte de extensiones analógicas, bien para terminales telefónicos analógicos, terminales de Telecomunicaciones Inalámbricas Digitalmente Mejoradas (Digital Enhanced Cordless Telecommunications, DECT por sus siglas en inglés) o bien equipos de fax.

- Soporte de líneas y extensiones IP: SIP, H323 o IAX, SCCP, MGCP.
- Música en espera basada en archivos MP3 y similar.
- Itinerancia, llamada en su conmutador desde cualquier ubicación.
- Integración con bases de datos.
- Sistema de menú en Pantalla ADSI (Interfaz Analógico para presentación de Servicios).
- Receptor de alarmas Agregar Mensaje (Append Message).
- Autentificación de llamadas con respuesta automatizada.
- Opciones de registros de llamada detallados.
- Sistema de grabación de llamadas.
- Recuperación de llamadas (DID y ANI).
- Configuración de llamada en espera.
- Configurable para trabajar con conferencia de voz.
- Opciones de marcado por nombre, marcación predictiva.
- Acceso directo al sistema interno.
- Empleo de agentes locales y remotos.
- Opciones de privacidad.
- Funciones básicas de Usuario:
    - Transferencia directa o consultiva.

- Llamada directa a extensión.
- Retrollamada (Callback).
- DND (No Molestar).

2.1.2 Beneficios de Asterisk

Funcionalidad: Asterisk dispone de muchas funcionalidades, desde las más básicas hasta las más avanzadas.

Escalabilidad: El sistema puede dar servicios hasta 10.000 usuarios.

Competitividad en coste: Por ser un software de código abierto y utilizar una plataforma servidor estándar con tarjetas PCI para las interfaces de telefonía.

Interoperabilidad y Flexibilidad: Incorpora la mayoría de estándares de telefonías del mercado, lo que le permite conectarse a la PSTN e integrarse fácilmente con centralitas IP.

2.2 Tarjetas de Telefonía FXO/FXS

Se conoce con el nombre de FXO y FXS a los puertos usados por las líneas telefónicas analógicas (también denominados Servicio Telefónico Básico y Antiguo – POTS) [4].

2.2.1 Interfaz de Central Externa FXO

Es el puerto que se comunica con la central telefónica o PBX. Sólo reciben tono de marcado desde las líneas (FXS). Son más conocidos como las entradas de las troncales, una aplicación de estos puertos se muestra en la siguiente figura 2.2.

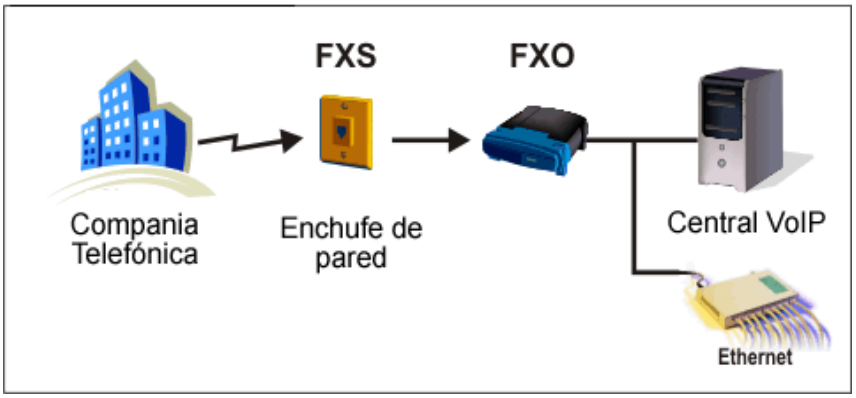

Fig. 2.2 Conexión FXO a una central PBX.

2.2.2 Interfaz de Abonado Externo FXS

Es el puerto que permite simular el comportamiento de una línea telefónica, pues envía tono de marcado, corriente para la

batería y tensión de llamada. Se las conoce como salidas de extensión y su utilización se observa en la figura 2.3.

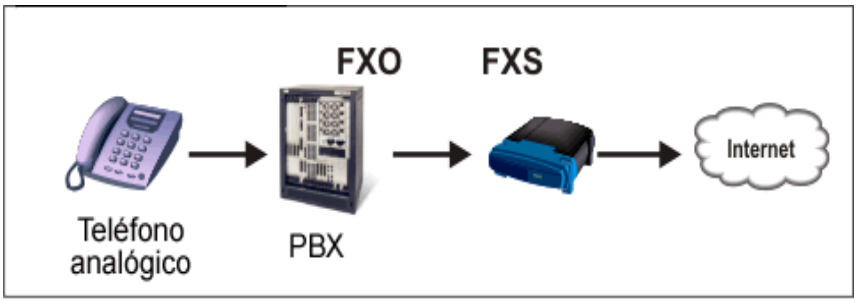

Fig. 2.3 Conexión FXS a una central PBX.

2.3 DAHDI, Interfaz de Dispositivos Digium de Asterisk

DAHDI brinda soporte para hardware, además de proporcionar controladores para tarjetas. Hasta el 19 de Mayo del 2008 era conocido con el nombre de Zaptel. Este paquete de DAHDI está compuesto por dos subpaquetes que son:

Kernel: Incluye los módulos y firmwares.

Tools: Sirven para controlar los canales, entre las herramientas más importantes encontramos: dahdi_gencof, dahdi_scan, dahdi_tool, entre otras.

Para configurar DAHDI se deben realizar cambios en los archivos:

/etc/dahdi/system.conf

/etc/asterisk/chan_dahdi.conf [5]

2.4 AMI, Interfaz de Administración de Asterisk

La interfaz de administración de Asterisk permite que programas clientes se conecten a Asterisk mediante el puerto TCP/IP: 5038, que es el que se utiliza por defecto. Una vez conectados y autenticados pueden comenzar a ejecutar comandos y leer eventos, lo que proporciona la posibilidad de alterar el comportamiento de Asterisk desde un programa hecho a medida.

La nueva interfaz gráfica de Asterisk funciona con AJAM (Administración de Asterisk con Javascript Asíncrono), que se conecta con Asterisk mediante HTTP. AMI es una interfaz conceptualmente similar a AGI, pero su incumbencia es exclusivamente la de administración [6].

2.4.1 Tipos de Paquetes

Existen 3 tipos de paquetes que vienen determinados por las siguientes claves:

Action: Paquete enviado desde el programa cliente hacia Asterisk, haciendo el requerimiento de que se ejecute una acción en particular. Estas acciones dependen de los módulos presentes en el servidor Asterisk y sólo puede realizarse una acción a la vez.

Response: Paquete enviado desde Asterisk hacia el cliente como respuesta a la última acción realizada.

Event: Son los datos pertinentes al evento generado por el servidor Asterisk o un módulo.

2.4.2 Configuración de AMI

Como se mencionó anteriormente se debe habilitar el puerto 5038 y adicionalmente configurar usuarios en el archivo manager.conf, el mismo que se encuentra ubicado en el directorio /etc/asterisk/. A cada uno de estos usuarios se le establecen un conjunto de IPs permitidas, así como negadas y ciertos permisos, entre ellos: system, verbose, user, config, call, originate, comand, entre otros.

Aunque se pueden manejar muchos programas cliente con un mismo usuario, es preferible tener un usuario para cada aplicación, las mismas que pueden ser desarrolladas en diferentes lenguajes de programación como son: .NET, C++, C#, PHP, Phyton, Ruby y Perl.

CAPÍTULO 3

3. IMPLEMENTACIÓN

3.1 Hardware

Previo al establecimiento de las especificaciones de hardware se deben considerar cuidadosamente dos aspectos importantes: el diseño general del sistema y las funcionalidades que éste tendrá, esto ayudará a determinar la marca y modelo del CPU, tarjeta madre y fuente de energía.

3.1.1 Servidor

El equipo donde se han configurado los diferentes servidores que se emplearán en el desarrollo de este proyecto tiene las características descritas en la Tabla I.

Procesador	Intel Core 2 Quad
RAM	3 GB
Disco Duro	80 GB
Tarjeta de Red	10/100 Mbps

Tabla I - Características del Servidor

3.1.2 Tarjeta Analógica TDM410p

Las características de la tarjeta que se utilizará para la implementación del proyecto se describen en la Tabla II.

Nombre	TDM410P
Puertos	4
Tamaño	165*106*18 mm
Peso	240 gr

Tabla II - Características de la Tarjeta Analógica TDM410p

3.2 Software

3.2.1 Servidor PBX: Asterisk

El servidor que será utilizado como centralita telefónica tendrá instalados los componentes que se detallan en la Tabla III.

Sistema Operativo	Centos versión 5.4
Software IP PBX	Asterisk versión 1.6.2.17.2

Tabla III - Servidor PBX

Librerías necesarias para que Asterisk funcione correctamente como una PBX.

- kernel
- kernel-devel
- gnutls-devel
- bison
- bison-devel
- gcc
- ncurses
- ncurses-devel
- gcc-c++
- zlib
- zlib-devel
- openssl
- openssl-devel

3.2.2 Servidor de Base de Datos: MySQL

El servidor que será utilizado para administrar la base de datos proporcionada por la organización será: mysql-5.0.77-4.el5_5.5.

3.2.3 Servidor Web: Apache

El servidor que será utilizado para generar la página de administración del sistema de cobranza será: httpd-2.2.3-31.el5.centos.

3.2.4 Paquetes adicionales

- Para crear el archivo de audio, que escuchará el cliente, se utilizará el paquete de conversión de texto a voz: festival-1.95-5.2.1 [7].

- Para cambiar el formato del archivo generado por Festival, se utilizará el paquete Sound eXchange: sox-12.18.1-1.el5_5.1 [8].

3.3 Instalación

3.3.1 Instalación de Librerías Base

Mediante el paquete administrador de software YUM se instalan o actualizan los paquetes del sistema operativo.

El *–devel* indica que se instalarán paquetes adicionales a kernel, bison, ncurses, openssl. Estos paquetes *-devel*

incluyen las cabeceras y librerías necesarias para desarrollar aplicaciones con cada uno de ellos.

Dentro de la consola de CentOS ejecutar el siguiente comando:

```
yum install kernel kernel-devel bison bison-devel ncurses ncurses-devel openssl openssl-devel  libtermcap-devel gcc gcc-c++
```

3.3.2 Instalación de Asterisk

Se ingresa a la página de descargas de Asterisk[9], y se procede a realizar la descarga de los paquetes comprimidos Asterisk y Dadhi. Para continuar con la instalación se deben descomprimir los paquetes descargados, de la siguiente manera:

Se accede al directorio donde se descargaron los paquetes con el comando

```
cd /usr/src
```

Se descomprimen los archivos

```
tar -xzvf asterisk-1.6.2.17.2.tar.gz
```

```
tar –xzvf dahdi-linux-complete-
2.4.1.2+2.4.1.tar.gz
```

Se accede al directorio donde se descomprimió dahdi-linux-complete-2.4.1.2+2.4.1

```
cd dahdi-linux-complete-2.4.1.2+2.4.1
```

Se instala el módulo DAHDI ejecutando los siguientes comandos secuencialmente

```
make clean
make
make install
make config
```

Luego se procede a instalar Asterisk, para lo cual se accede al directorio donde se descomprimió asterisk-1.6.2.17.2

```
cd ../asterisk-1.6.2.17.2
make clean
./configure
make menuselect
make
make install
```

```
make samples
make config
```

Una vez terminada la instalación automáticamente se crea una carpeta llamada asterisk en el directorio /etc. De esta manera los archivos de configuración de Asterisk se encuentran bajo el directorio /etc/asterisk/ [10].

3.3.3 Instalación de Servicios Adicionales

Estos servicios adicionales, que han sido instalados a través de YUM, son necesarios para el desarrollo del sistema.

- Base de Datos MySQL

```
yum install -y mysql mysql-devel mysql-server
```

- Servidor Web Apache

```
yum install -y httpd
```

- Aplicación de conversión de texto a voz Festival

```
yum install -y festival festival-devel
```

- Convertidor de Archivos de Audio SOX

```
yum install -y sox
```

3.4 Configuración de archivos de Asterisk

3.4.1 Configuración del archivo manager.conf

Este archivo se encuentra en la ruta /etc/asterisk/manager.conf, y contiene las configuraciones del AMI (Asterisk Manager Interface).

En la sección [general] se habilita al administrador para que “escuche” los requerimientos del programa cliente, el puerto por defecto es el 5038.

```
[general]
enabled=yes
port=5038
bindaddr=0.0.0.0
webenabled=no
```

Luego se crean usuarios, siendo el nombre de los mismos el definido entre corchetes y la contraseña el valor configurado con la opción secret. Con estos valores se podrá establecer la conexión con el programa cliente. Es necesario habilitar ciertos permisos (read para lectura y write para escritura),

según las funcionalidades que se le quieran dar a la aplicación.

Para realizar la conexión a AMI se han creado cuatro usuarios con el siguiente formato:

```
[outboundCallX]
secret=secretpass
deny=0.0.0.0/0.0.0.0
permit=127.0.0.1/255.255.255.255
read=system,call,log,verbose,command,agent,user,originate
write=system,call,log,verbose,command,agent,user,originate
```

3.4.2 Configuración del archivo extensions.conf

Este archivo se encuentra en la ruta /etc/asterisk/extensions.conf, y contiene el plan de marcado de la central telefónica.

El archivo extensions.conf es el más importante de Asterisk y tiene como objetivo principal definir el plan de marcado, que determina el comportamiento que seguirá la central telefónica.

El fichero extensions.conf se compone de secciones o contextos definidos entre corchetes [ ]. Existen dos contextos especiales, [general] en el que se definen reglas generales para los demás contextos del dialplan y [globals] en el que se definen variables globales. La utilización de este último contexto es opcional, no siendo así el de [general].

Contexto [salientes]

Una vez que la llamada ha sido establecida se direcciona a este contexto.

En la extensión 200 de este contexto se reproduce el archivo que contiene los datos del pago a realizar por cada cliente. Seguido se reproduce un nuevo archivo de audio en el que se le da a conocer las opciones de los dígitos que puede presionar:

- 5, si escucha el mensaje completo, lo cual lo llevará a la extensión 5 dentro del mismo contexto, en la que se cambiará el estado de la llamada a realizada (estado_llamada=1) y se dará por terminada la llamada.

- 3, si desea escuchar nuevamente el mensaje, con lo que será conducido a la extensión 3, que lo redirigirá a la extensión 200 del contexto.

Si presiona un dígito diferente a los descritos anteriormente, se direccionará a las opciones del usuario.

En la figura 3.3, se presenta el diagrama de flujo del comportamiento del plan de marcado.

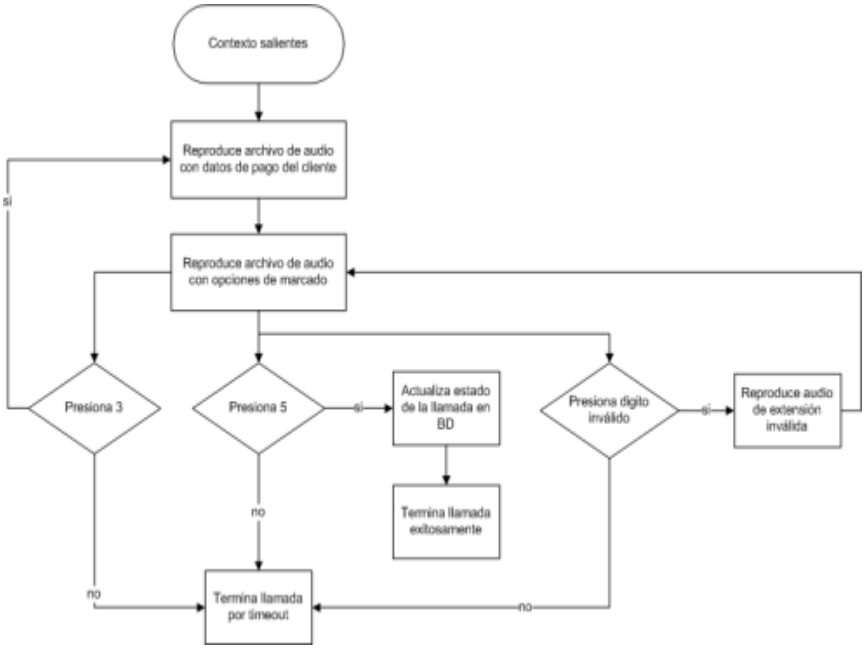

Fig 3.3 Diagrama de flujo de extensions.conf

```
[general]
autofallthrough=yes
clearglobalvars=no

[salientes]
exten => 200,1,Playback(outboundCall/archivoAudio${AUDIO})
exten => 200,n,SayUnixTime(${FECHA},,ABdY)
exten => 200,n,Background(outboundCall/opcion)
exten => 200,n,WaitExten(7)

;El cliente presiona un digito inválido
exten => i,1,Playback(outboundCall/invalido)
exten => i,n,Goto(200,3)

;El cliente desea escuchar nuevamente el mensaje
exten => 3,1,Goto(200,1)

;El cliente escuchó el mensaje completo
exten => 5,1,MYSQL(Connect connid localhost root password
base_clientes)
exten => 5,n,MYSQL(Query resultid ${connid} UPDATE cliente
SET estado_llamada=1 where no_cuenta=${IDCLIENTE})
exten => 5,n,MYSQL(Disconnect ${connid})
exten => 5,n,Playback(outboundCall/despedida)
```

```
exten => 5,n,Hangup(98)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()
```

3.4.3 Configuración del archivo system.conf

Este archivo se encuentra en la ruta /etc/dahdi/system.conf, y en éste se configura la señalización que utilizan los puertos presentes en la tarjeta analógica.

Es autogenerado al ejecutar el comando dahdi_genconf en la consola.

```
# Autogenerated by /usr/sbin/dahdi_genconf on Fri Jun  3 09:56:02
2011
# If you edit this file and execute /usr/sbin/dahdi_genconf again,
# your manual changes will be LOST.
# Dahdi Configuration File
# This file is parsed by the Dahdi Configurator, dahdi_cfg
# Span 1: WCTDM/0 "Wildcard TDM410P Board 1" (MASTER)
fxsks=1
echocanceller=mg2,1
fxsks=2
```

```
echocanceller=mg2,2
fxsks=3
echocanceller=mg2,3
fxsks=4
echocanceller=mg2,4

# Global data
loadzone      = us
defaultzone  = us
```

3.4.4 Configuración del archivo chan_dahdi.conf

Este archivo se encuentra en la ruta /etc/asterisk/chan_dahdi.conf, y en éste se configuran los puertos dahdi con que cuenta la tarjeta analógica.

```
[channels]
; canales físicos: opciones por defecto para todos los canales
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
```

```
callprogress=yes
busydetect=yes
busycount=4
language=es

; definicion de canales:
group=0
context=haciaPSTN
signalling=fxs_ks ; señalizacion FXS para canal FXO
channel => 1-4        ; PSTN se conecta al puerto 4
```

3.4.5 Configuración del archivo sip.conf

Este archivo se encuentra en la ruta /etc/asterisk/sip.conf, y en éste se configuran todos los usuarios de tipo SIP.

En este archivo se pueden observar dos secciones principales, la sección [general] en la que se definen las opciones que todos los usuarios/peers van a seguir, y la sección de los usuarios [nombre_usuario] en la que se definen uno a uno los usuarios con opciones más específicas.

A continuación se detalla la configuración que ha sido implementada para la realización de las pruebas iniciales de este proyecto:

```
[general]
context=noautenticadas
allowguest=no
srvlookup=yes
udpbindaddr=0.0.0.0
tcpenable=no
qualify=yes
language=en

[usuarioX]
type=friend
context=internos
host=dynamic
nat=yes
secret=ast3r1sk
dtmfmode=auto
disallow=all
allow=gsm
;allow=ulaw
callerid="usuarioX" <20X>
```

3.5 Scripts y página PHP

3.5.1 Script de generación de llamadas

Los siguientes scripts: amiCanal1.php, amiCanal2.php, amiCanal3.php, amiCanal4.php, ami2Canal1.php, ami2Canal2.php, ami2Canal3.php, ami2Canal4.php, ubicados en la ruta /etc/asterisk/outboundCall/, son los que se conectan con la Interfaz de Administración de Asterisk y obtienen los datos de la base para crear los archivos de audio que serán reproducidos a los clientes.

AmiCanal#.php se encargan de realizar las llamadas a todos los clientes, divididos en cuatro grupos, cada grupo asignado a un canal diferente; en caso de que hubieran clientes que no contesten, son ami2Canal#.php los que llamarán a los clientes que no pudieron ser contactados la primera vez.

En la figura 3.4, se presenta el diagrama de flujo del algoritmo y en el Anexo A la implementación del mismo desarrollado en lenguaje PHP.

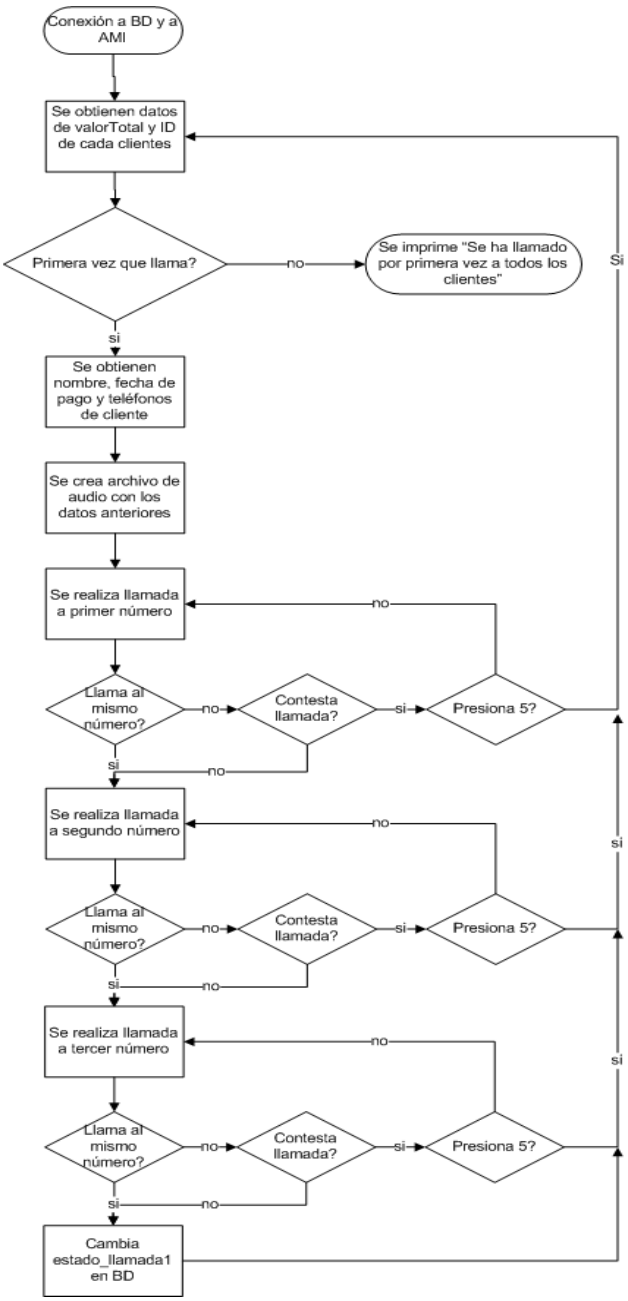

Fig 3.4 Diagrama de flujo de scripts amiCanal#.php, ami2Canal#.php

3.5.2 Script de la clase Llamada

El siguiente script, ubicado en la ruta /etc/asterisk/outboundCall/, es una clase que incluye dos funciones:

- realizarLlamada, genera la llamada mediante el Action:Originate haciendo uso de los parámetros que se le pasan. Devuelve 0 si la llamada se realizó con éxito o 1 si no fue así.

- PresionaCinco, devuelve 1 si el cliente presionó 5 al escuchar el mensaje o 0 si no fue así.

La implementación en PHP de este script se muestra en detalle en el Anexo B.

3.5.3 Aplicación Web

El siguiente script, ubicado en la ruta /var/www/html/, es la aplicación web (programa cliente) desde la cual se ejecutan los scripts amiCanal#.php, ami2Canal#.php y generaInforme.php.

La implementación en PHP de este script se muestra en detalle en el Anexo C.

3.5.4 Script que genera informe de llamadas

El siguiente script, ubicado en la ruta /var/www/html/, utiliza la clase class.ezpdf.php para generar un informe, en formato pdf, de las llamadas realizadas a todos los clientes. Este informe será entregado a la organización.

La implementación en PHP de este script se muestra en detalle en el Anexo D.

CAPÍTULO 4

4. FUNCIONAMIENTO Y PRUEBAS

Previo a la realización de las pruebas se procede a instalar un segundo servidor Asterisk, que simula la PSTN. Luego de la instalación de los componentes necesarios en los servidores, se realizan las conexiones entre los mismos, así como de los teléfonos al servidor PSTN.

También es necesario asegurar que los servicios de Base de Datos (MySQL) y Web (Apache), estén ejecutándose en la máquina, en caso de que no fuese así, se pueden levantar los servicios con los siguientes comandos:

```
service mysqld start
service httpd start
```

4.1 Inicialización e ingreso a Asterisk

Los siguientes son comandos que permiten iniciar o detener el servicio de Asterisk.

`service asterisk start`	Iniciar servicio.
`service asterisk stop`	Detener servicio.
`service asterisk status`	Obtener estado del servicio.
`service asterisk restart`	Reiniciar servicio.

Los siguientes comandos permiten ingresar y salir de la consola remota de asterisk así como también ejecutar acciones sin necesidad de ingresar a la misma.

`asterisk –c`	Iniciar Asterisk y abrir la consola remota
`asterisk –r`	Ingresar a la consola remota
`stop now`	Detener el servicio Asterisk desde la consola remota.
`exit`	Salir de la consola remota. No detiene el servicio Asterisk.

4.2 Conexión a AMI desde consola

En esta prueba se verificará que es posible establecer una conexión con la Interfaz de Administración de Asterisk, para cumplir con este fin se realizaron los siguientes pasos:

a. Hacer telnet desde la consola del sistema operativo al puerto configurado en el archivo *manager.conf* .
b. Enviar paquete con el *Action* de autenticación.
c. El resultado esperado es un mensaje de conexión exitosa.

Ejecución de la prueba:

a. Desde la consola se escribió el comando `telnet 127.0.0.1 5038` como se muestra en la figura 4.1:

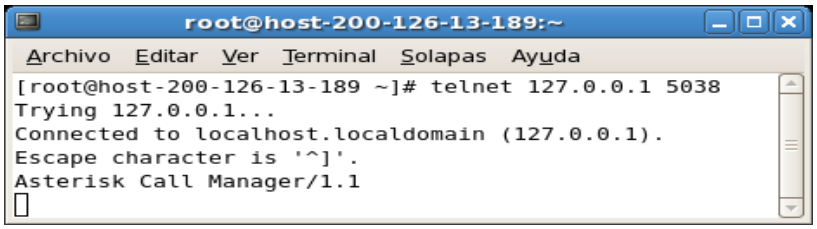

Fig 4.1 Conexión al puerto 5038 a través de telnet.

b. Se creó el paquete a enviarse con las siguientes características, como se ve en la figura 4.2:

```
Action: Login

UserName: <usuario>

Secret: <claveUsuario>
```

Fig 4.2 Envío de paquete para realizar conexión a AMI.

c. El resultado de la conexión se puede apreciar en la figura 4.3:

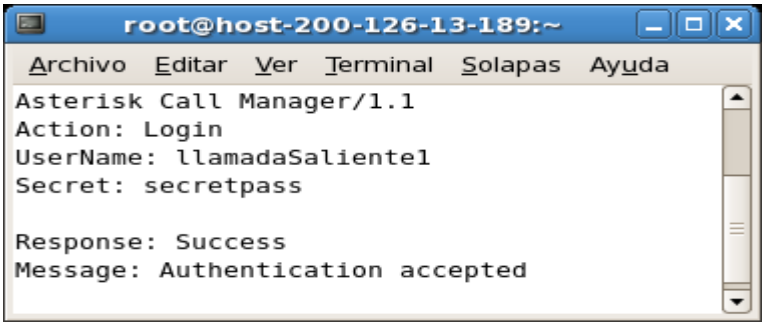

Fig 4.3 Resultado exitoso de la conexión a AMI.

4.3 Simulación de PSTN

Se requiere que el sistema implementado realice llamadas a números conectados a la PSTN, debido a que no se contó con los recursos para realizar estas llamadas, se instaló un nuevo servidor Asterisk que simula la red telefónica.

4.3.1 Comunicación entre puertos FXS

Dentro del plan de marcado de este nuevo servidor, se configuran 8 extensiones, cada una de las cuales son asignadas a los puertos de la tarjeta. La finalidad de esta configuración es simular la lógica de una PSTN, es decir que se genera una llamada desde un teléfono y ésta es recibida por otro. Lo anteriormente descrito se logra ejecutando los siguientes pasos:

a. Configurar el extension.conf del segundo servidor.
b. Conectar los teléfonos analógicos a los puertos de la tarjeta.
c. Realizar una llamada de prueba.

Ejecución de la prueba:

a. Se configuraron 8 extensiones, como se visualiza en la figura 4.4, las mismas que fueron asignadas a cada uno de los puertos. En la consola de Asterisk, se ejecutó el comando `dialplan reload`, para recargar el plan de marcado.

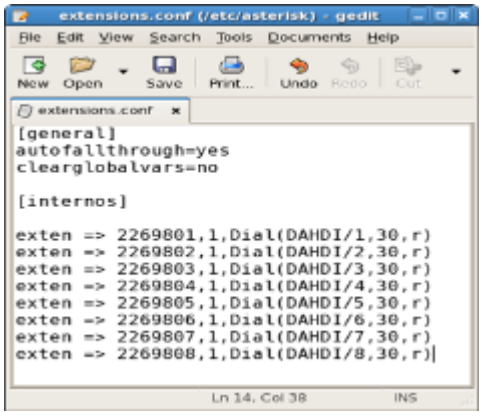

Fig 4.4 Archivo extensions.conf

b. Se realizó una llamada desde el teléfono conectado en el puerto 7 hacia el teléfono conectado en el puerto 8 de la tarjeta de telefonía. En la figura 4.5 se observa en la consola de Asterisk, la comunicación entre los dos canales antes mencionados.

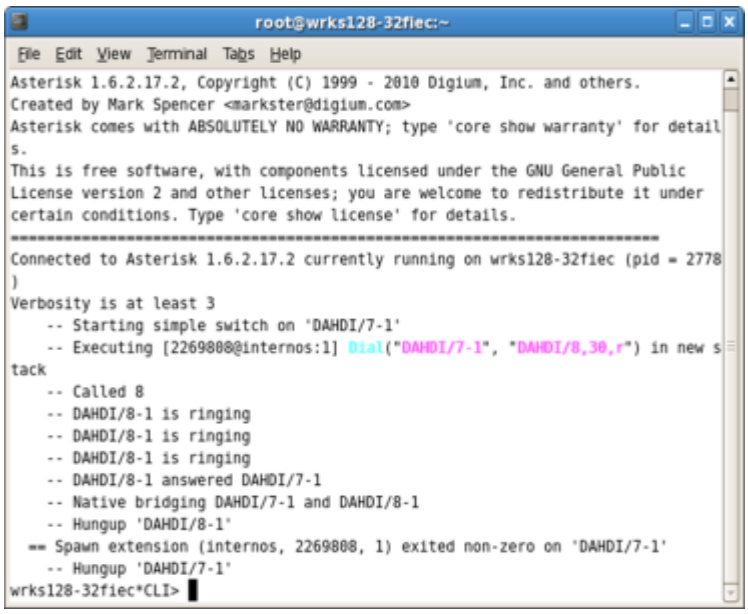

Fig 4.5 Comunicación entre puerto 7 y 8 de la tarjeta de telefonía.

4.4 Generación de llamadas

Con esta prueba se pretende generar llamadas desde una aplicación cliente, lo que se logrará a través de la ejecución de los scripts amiCanal#.php y ami2Canal#.php, los cuáles se conectan a la Interfaz de Administración de Asterisk, a la base de datos de la empresa y generan las llamadas. Para cumplir con este fin se realizaron los siguientes pasos:

a. Dentro de la consola, ubicarse en el directorio donde se encuentran los scripts.
b. Se le asigna permisos de ejecución a los scripts.
c. Se ejecuta el script.

Ejecución de la prueba:

a. Desde consola se escribió el comando

```
cd /etc/asterisk/outboundCall/
```

b. Se ejecutó el script mediante el comando

```
php amiCanal#.php
```

c. AMI genera eventos con los cuales se pudo monitorear la ejecución de las llamadas. En la figura 4.6 se exponen los eventos que se generaron al realizar una llamada.

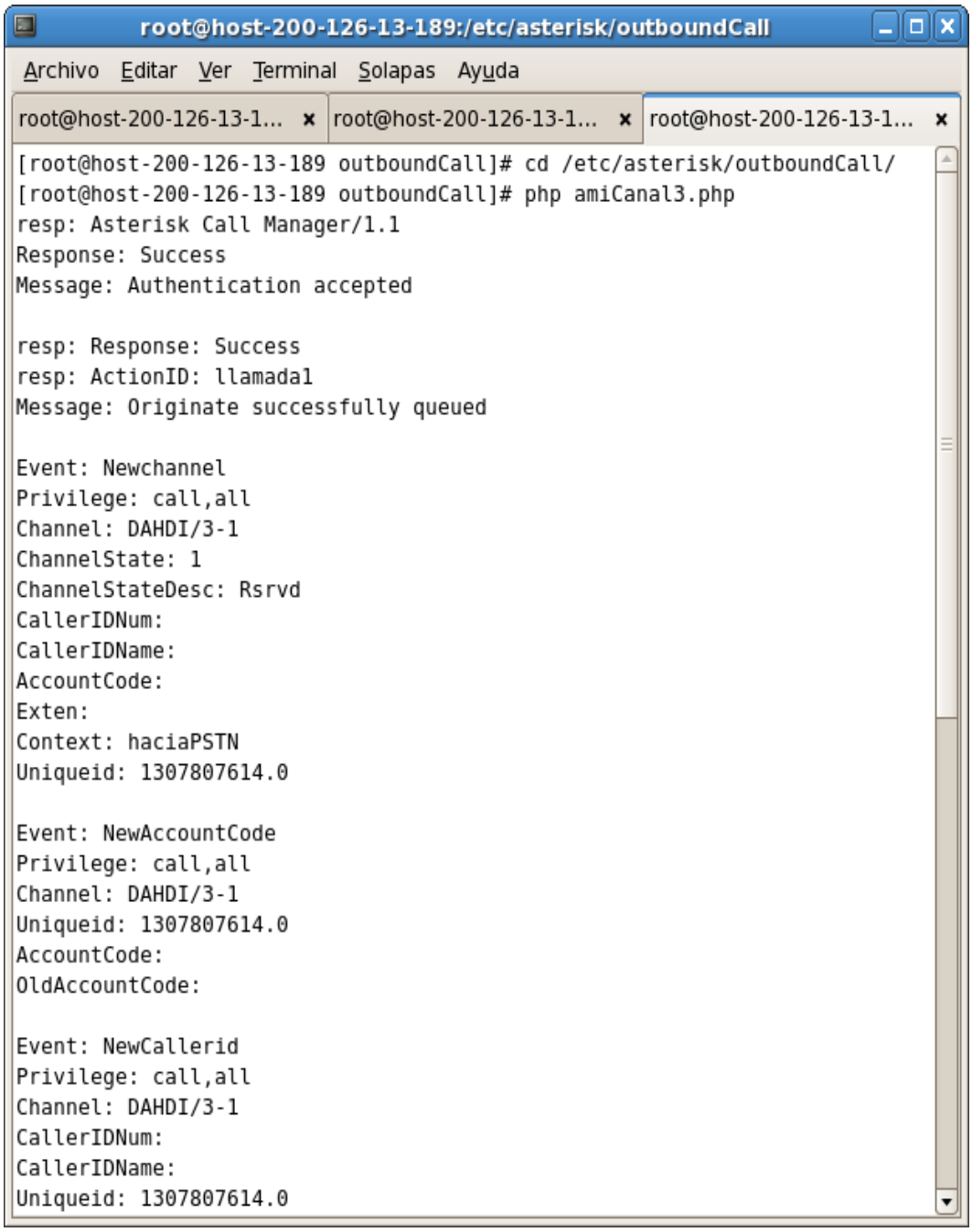

Fig. 4.6 Eventos generados por AMI al realizar un Action: Originate

Al ejecutar el script amiCanal#.php o ami2Canal#.php, se realizan las llamadas, las mismas que podrían ser exitosas o no.

4.4.1 Llamada exitosa

Al lograr establecer la comunicación con el cliente se pueden presentar los siguientes casos:

Caso 1: Si el cliente cuelga la llamada antes de escuchar el mensaje completo, se llama al mismo número únicamente por una segunda ocasión. En la figura 4.7 se indica como la primera llamada resulta no exitosa, y se genera una nueva llamada al mismo número.

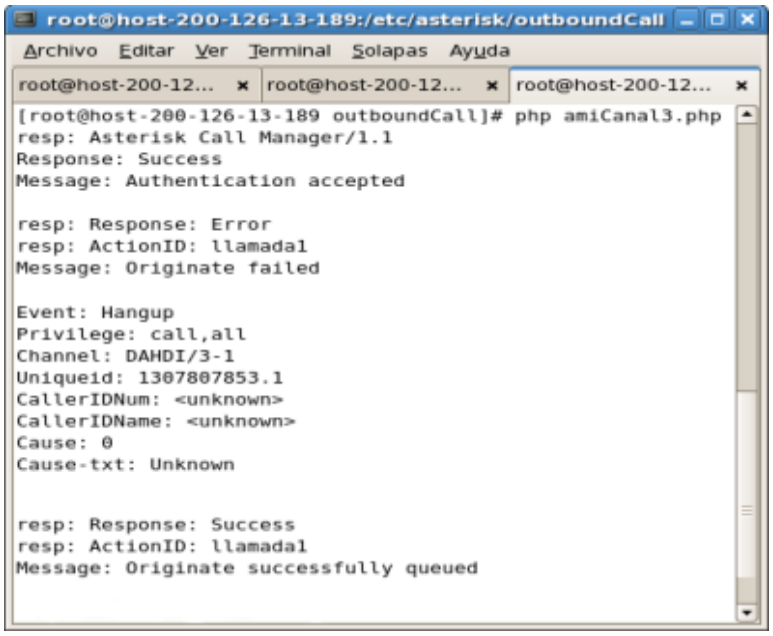

Fig. 4.7 Eventos generados por AMI al realizarse una llamada exitosa y colgarse.

Caso 2: Si el cliente al escuchar las opciones presiona 3, vuelve el escuchar el archivo de audio con sus datos. En la figura 4.8 se visualiza la salida en la consola de Asterisk.

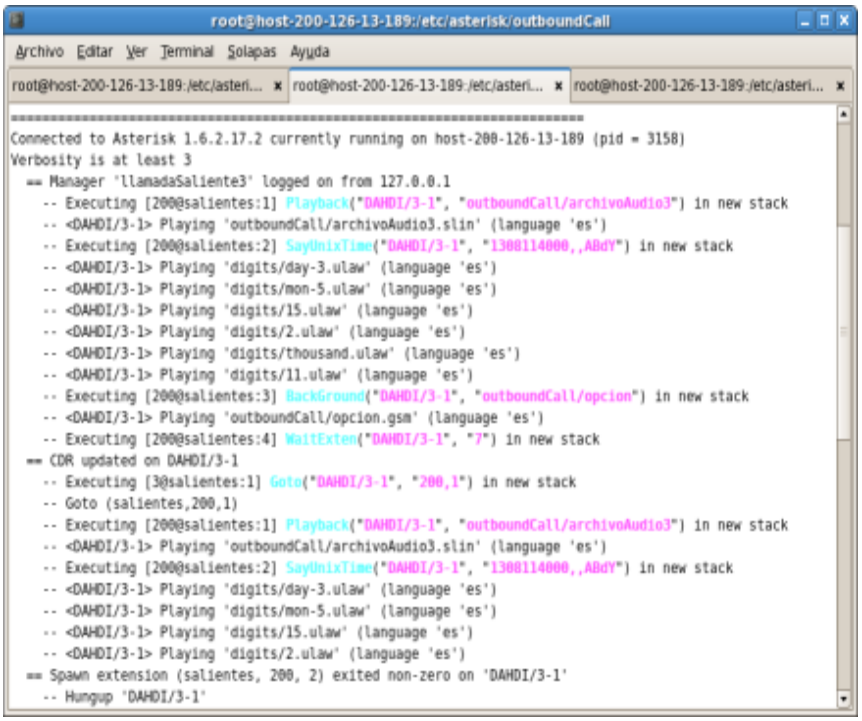

Fig. 4.8 Salida en la consola de Asterisk al presionar la tecla 3.

Caso 3: Si el cliente al escuchar las opciones presiona 5, se valida que escuchó el mensaje completo, por lo que se procede a actualizar los datos en la base. En la figura 4.9 se aprecia la salida en la consola de Asterisk.

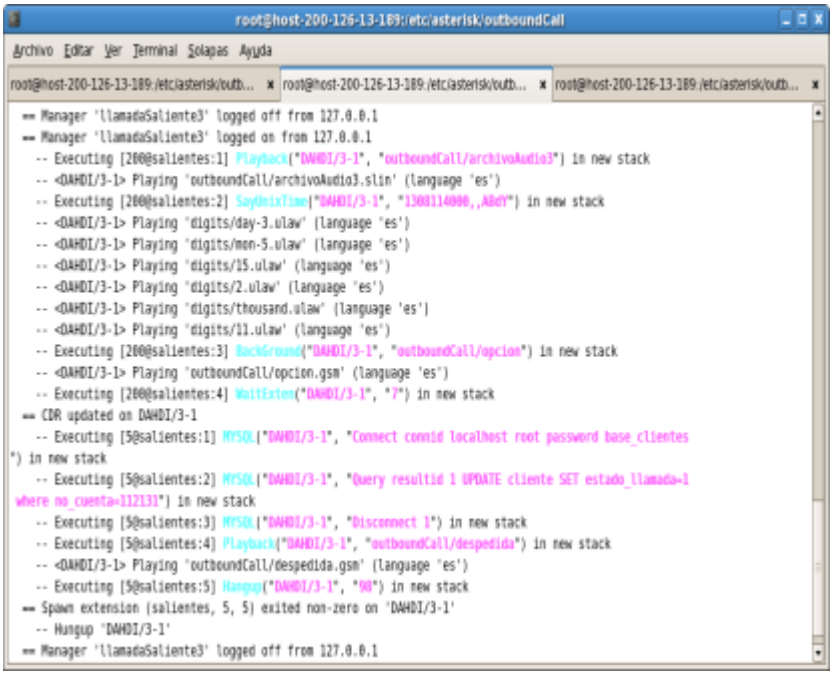

Fig. 4.9 Salida en la consola de Asterisk al presionar la tecla 5.

Caso 4: Si presiona un dígito diferente a los anteriores, volverá a escuchar las opciones del plan de marcado. En la figura 4.10 se muestra la salida en la consola de Asterisk.

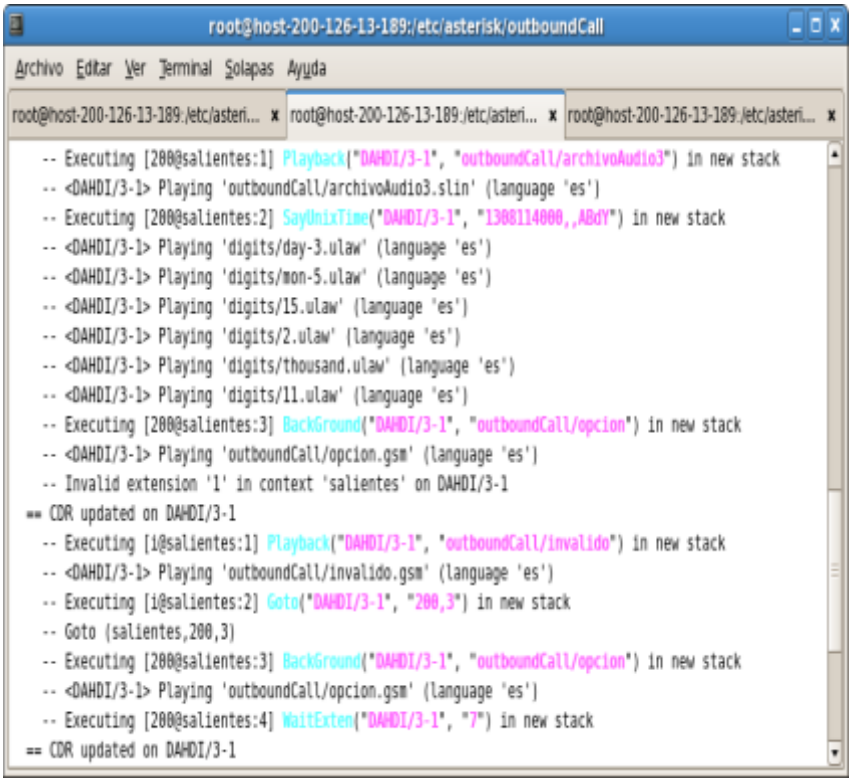

Fig. 4.10 Salida en la consola de Asterisk al presionar una tecla diferente de 3 o 5.

4.4.2 Llamada fallida

Al no poder realizarse la llamada, el sistema marca al siguiente número del cliente. En la figura 4.11 se indica como la primera llamada resulta no exitosa, y se genera la llamada al siguiente número del cliente.

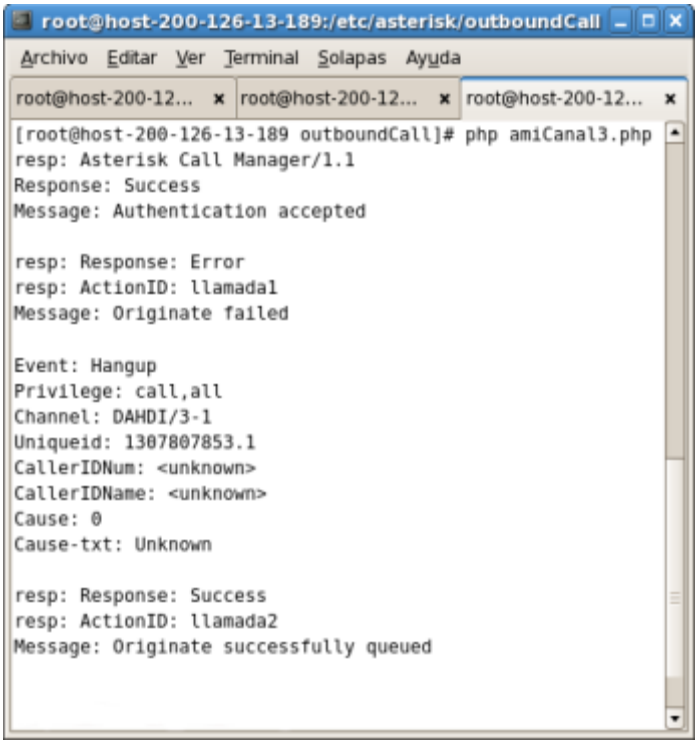

Fig. 4.11 Reporte de eventos generado por el AMI al realizarse una llamada no exitosa.

4.5 Generación simultánea de llamadas.

En esta prueba se verificará que las llamadas se realizan simultáneamente a los diferentes clientes, una llamada por cada canal disponible. Para alcanzar este objetivo se realizaron los siguientes pasos:

a. Desde el navegador, abrir la aplicación web.
b. Presionar el botón correspondiente a la acción que se requiera, ya sea realizar las llamadas por primera vez, o al segundo grupo de llamadas.
c. Se empiezan a realizar las llamadas por los 4 canales.

Ejecución de la prueba:

a. En el navegador web, se escribió la dirección http://localhost/empresaCobranza/administradorCobranza.php , como se muestra en la figura 4.12.

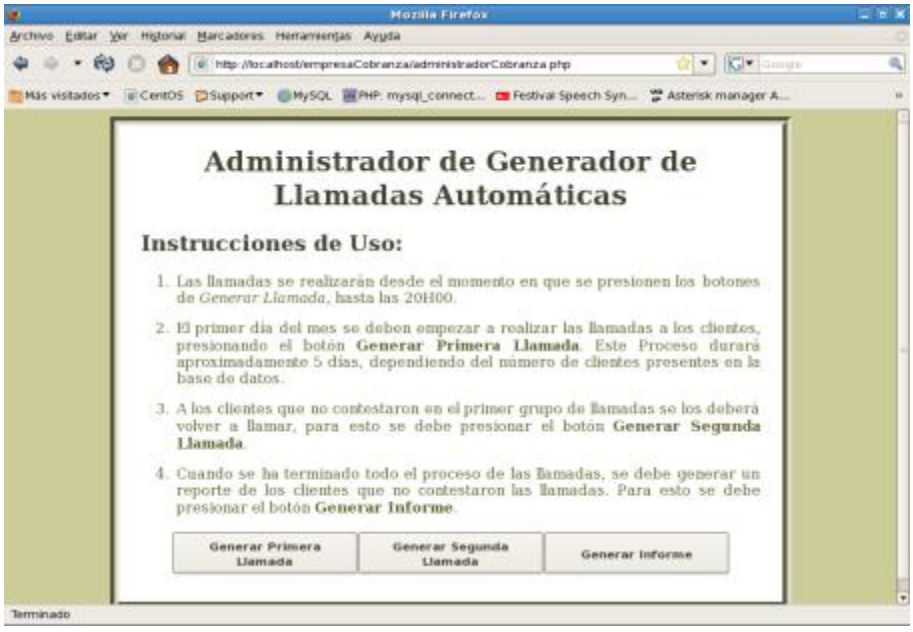

Fig. 4.12 Pantalla de la aplicación Empresa de Cobranzas.

b. Se presionó el botón Generar primera Llamada (el botón Generar segunda Llamada realiza la misma acción, pero para el segundo grupo de llamadas).

c. Se abrió un cuadro de diálogo en el que apareció un mensaje que indica que las llamadas empezaron a realizarse, como puede apreciarse en la figura 4.13.

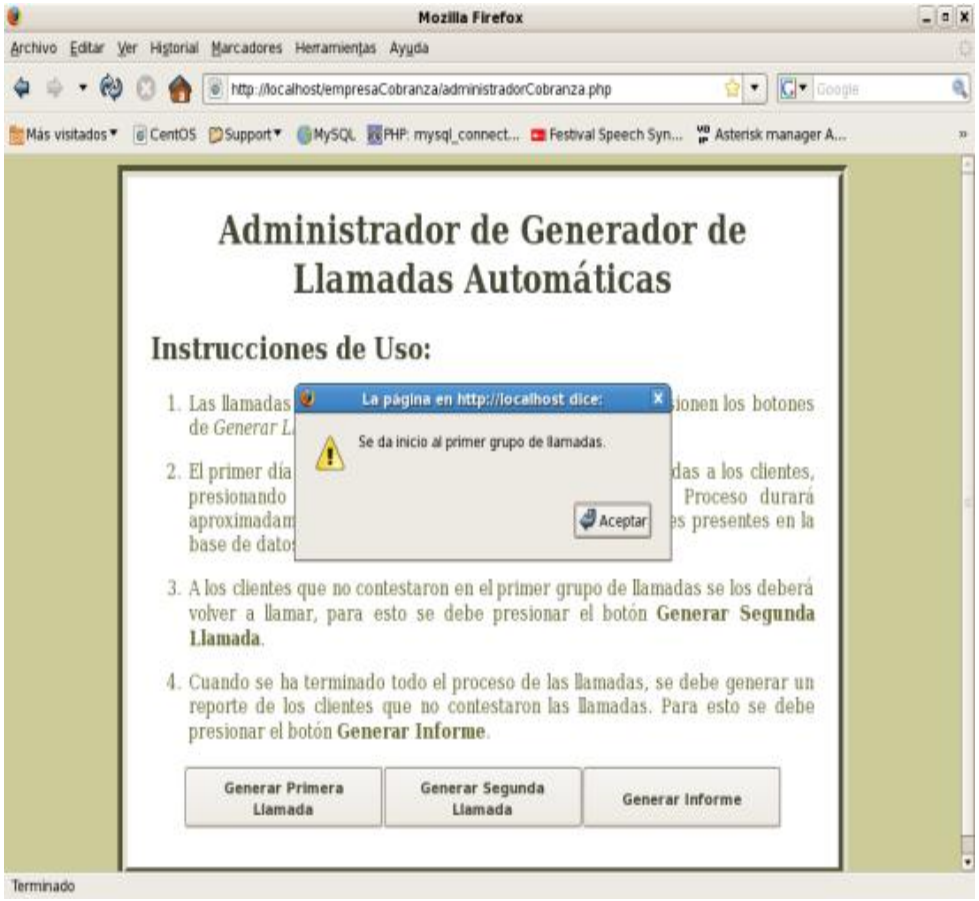

Fig. 4.13 Cuadro de diálogo inicio de llamadas

4.6 Generación del reporte de llamadas.

En esta prueba se generará un reporte en formato pdf de las llamadas exitosas y no exitosas, que será entregado a la empresa contratista para que pueda tener un control más eficiente de los clientes contactados. Para lograr este cometido se realizaron los siguientes pasos:

a. Desde el navegador, abrir la aplicación web.
b. Presionar el botón correspondiente a la generación de reportes.
c. Se genera un reporte en formato pdf.

Ejecución de la prueba:

a. En el navegador web, se escribió la dirección http://localhost/empresaCobranza/administradorCobranza.php
b. Se presionó el botón Generar reporte.
c. Se creó un reporte con las llamadas contestadas y no contestadas, como se ve en la figura 4.14.

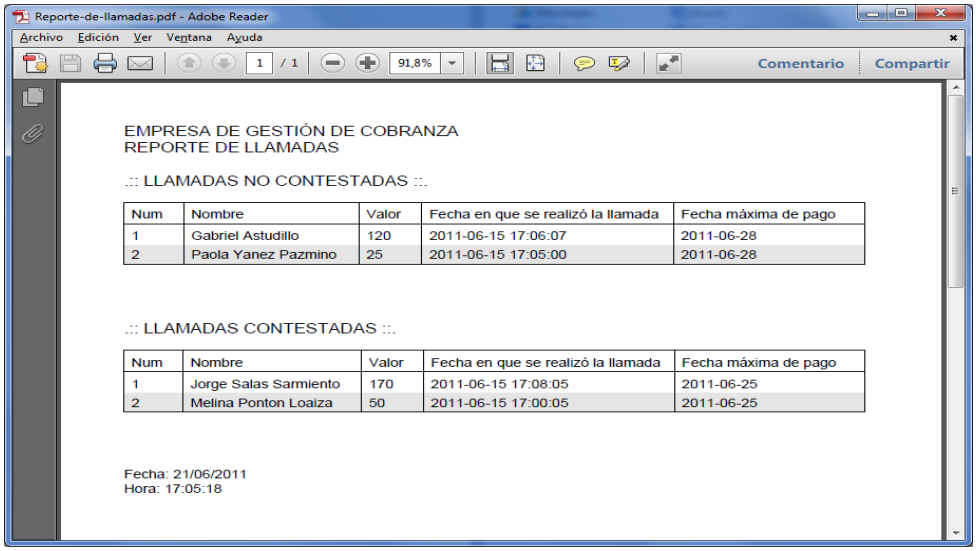

Fig 4.14 Reporte de Llamadas

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. La implementación de éste sistema garantiza la comunicación eficiente entre una empresa y sus clientes, dado que éstos pueden ser contactados a sus números de casa, celular u oficina. Ésta es una alternativa eficaz con respecto a la contratación de personal que realice dichas llamadas.

2. La central de código abierto Asterisk permite desarrollar aplicaciones escalables y modulares, ajustadas a las necesidades propias de quienes las requieran. A diferencia de una PBX tradicional, ésta se configura enteramente modificando archivos, mientras que en la tradicional se deben agregar módulos de hardware para cumplir este fin.

3. La interfaz gráfica implementada en el sistema es sencilla, amigable con el administrador del mismo y de fácil interacción, razón por la cual ha sido alojada en el mismo servidor de Asterisk, minimizando así los costos de inversión.

4. Aunque para el desarrollo del proyecto se utilizó como escenario de prueba una Empresa de Cobranzas, el sistema puede ser utilizado en otros tipos de negocios y para otros fines.

5. Se aprovechan en gran medida los recursos de la empresa al poder realizar llamadas simultáneas a diferentes clientes, pudiendo efectuar mayor número de llamadas en el día con la misma efectividad que si se contactara a un cliente a la vez.

6. El sistema está desarrollado para que se vuelva a llamar una vez más al mismo número, en caso de que la llamada sea interrumpida mientras se reproduce el mensaje. Esto puede ser fácilmente modificado de acuerdo a las necesidades de la empresa.

Recomendaciones

1. Determinar las características de los equipos servidores en función del número de usuarios y tipo de servicios.

2. Recordar cambiar el código de provincia para llamadas locales según la ciudad donde se quiera utilizar el sistema.

3. La información de los clientes que se mantiene en la base de datos, debe ser actualizada mensualmente.

4. Se recomienda como trabajo a futuro implementar casillas de buzón de voz para los clientes, de tal forma que, aun cuando no se pueda contactar con el cliente, se le pueda dejar un mensaje, que podrá ser revisado posteriormente.

   Para configurar el buzón de voz se puede elaborar un script que cree automáticamente las casillas, a partir de los datos de los clientes que se encuentran almacenados en la base. Se deben crear también los respectivos contextos dentro del archivo extensions.conf.

GLOSARIO

Código abierto: Open Source es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

H.323: Recomendación del International Telecommunication Union (ITU), que define los protocolos para proveer sesiones de comunicación audiovisual sobre paquetes de red. No garantiza una calidad de servicio, además es independiente de la topología de la red y admite pasarelas, permitiendo usar más de un canal de cada tipo (voz, vídeo, datos) al mismo tiempo.

Plan de Marcado: o dialplan es un software residente en algunos gateways y en casi todas las IP-PBX que permite determinar el tratamiento que debe darse a un número discado.

PBX: Private Branch eXchange. Sistema telefónico dentro de una organización que maneja las llamadas entre sus usuarios en líneas locales mientras permite que entre todos los usuarios compartan un número determinado de líneas telefónicas externas.

Script: es un conjunto de instrucciones, sentencias de control, variables y demás elementos de programación generalmente almacenadas en un archivo de texto (pude considerarse como un archivo de instrucciones o como un programa).

Softphone: es un software que hace una simulación de teléfono convencional por computadora, es decir, permite usar la computadora para hacer llamadas a otros softphones o a otros teléfonos convencionales.

VoIp: Voice Over Internet Protocol (Voz Sobre el Protocolo de Internet). Básicamente VoIP es un método por el cual tomando señales de audio analógicas del tipo de las que se escuchan cuando uno habla por teléfono se las transforma en datos digitales que pueden ser transmitidos a traves de internet hacia una dirección IP determinada.

BIBLIOGRAFÍA

[1] Elastix, Voz sobre ip en Ecuador, http://listas.asle.ec/pipermail/asociacion/2009-October/012083.html, fecha de consulta junio 2011.

[2] Creditperformancenews, En Ecuador aumenta el acceso al crédito durante el segundo trimestre, http://www.creditperformancenews.com/es/notas/2010-09-00/en-ecuador-aumenta-el-acceso-al-credito-durante-el-segundo-trimestre-ecuador/, fecha de consulta junio 2011.

[3] Wikipedia, Asterisk, http://es.wikipedia.org/wiki/Asterisk, fecha de consulta junio 2011.

[4] 3cx, ¿Qué significan los términos FXS y FXO?, http://www.3cx.es/voip-sip/fxs-fxo.php, fecha de consulta junio 2011.

[5] Blogs.digium.com, Zaptel project being renamed to DAHDI, http://blogs.digium.com/2008/05/19/zaptel-project-being-renamed-to-dahdi/, fecha de consulta junio 2011.

[6] Voip-info, AMI - Action y ejemplos, http://www.voip-info.org/wiki/view/Asterisk+manager+API, fecha de consulta junio 2011.

[7] The University of Edinburgh, Festival, http://www.cstr.ed.ac.uk/projects/festival/manual/, fecha de consulta junio 2011.

[8] Thegeekstuff, Sox, http://www.thegeekstuff.com/2009/05/sound-exchange-sox-15-examples-to-manipulate-audio-files/, fecha de consulta junio 2011.

[9] Asterisk, Asterisk Downloads, http://www.asterisk.org/downloads, fecha de consulta junio 2011.

[10] Van Meggelen J., Smith J. y Madsen L.,
Asterisk The Future of Telephony, Editorial O'Relly Media, 2008

ANEXOS

ANEXO A: Script de Generación de Llamadas

```
<?php
/* CLASES A UTILIZAR */
include_once("llamada.php");

/* INSTANCIAS*/
$llamada1 = new llamada();
$llamada2 = new llamada();
$llamada3 = new llamada();

/* VARIABLES */
$fail1=0; $fail2=0; $fail3=0;
$escuchoMsg=0; $sgteLlamada=1; $sgteLlamada1=1;
$detener=0;
$numReg=0; $numElse=0;
$i=0; $j=0; $k=0;
$codLocal="04";

/* CONEXIÓN A LA BASE */
$dbhost="localhost";
$dbuser="user1";
$dbpass="pass_user1";
$db="base_clientes";

$link= mysql_connect($dbhost,$dbuser,$dbpass);
```

```
if (!$link) {
die('No pudo conectarse: ' . mysql_error());
}

$db_selected = mysql_select_db($db, $link);
if (!$db_selected) {
die ('Can\'t use base_clientes : ' . mysql_error());
}

/* CONEXIÓN A MANAGER */
$mngrhost="127.0.0.1";
$mngruser="outboundCall1";
$mngrpass="secretpass";

$socket = fsockopen($mngrhost,"5038", $errno, $errstr,30);
fputs($socket, "Action: Login\r\n");
fputs($socket, "UserName: $mngruser\r\n");
fputs($socket, "Events: on\r\n");
fputs($socket, "Secret: $mngrpass\r\n\r\n");

/**** OBTENGO LOS DATOS PARA CREAR EL ARCHIVO DE AUDIO ****/
$queryMonto = "SELECT id_cliente, SUM(valor), COUNT(valor) FROM monto where
estado=0 GROUP BY id_cliente";
$resultMonto = mysql_query($queryMonto);
$numReg = mysql_num_rows($resultMonto);
$total = 0;
```

```
$numPagos =  0;

while($monto=mysql_fetch_assoc($resultMonto)){
$total = $monto["SUM(valor)"]; //obtengo suma de valores con estado =0
$numPagos = $monto["COUNT(valor)"]; //obtengo cuanto valores con estado =0
$idCliente = $monto["id_cliente"]; //obtengo id de c/cliente
$todosTls="";

$sgteLlamada=1;

/*Pregunto si ya se realizo la llamada*/
$estado=mysql_fetch_assoc(mysql_query("Select estado_llamada, estado_llamada1,
num_canal from cliente where no_cuenta=".$idCliente));

if($estado["estado_llamada"]==0 && $estado["estado_llamada1"]==0 &&
$estado["num_canal"]==4){

/*Obtengo los nombres y fechas de pago del cliente*/
$queryDatosCl = "SELECT nombre, fecha_pago FROM cliente where
no_cuenta=".$idCliente;
$resultDatosCl = mysql_query($queryDatosCl);
while($datosClA=mysql_fetch_assoc($resultDatosCl)){
$fecha=$datosClA["fecha_pago"];
$nombre=$datosClA["nombre"];
}
mysql_free_result($resultDatosCl);
```

```
/*Obtengo los nums de telefono del cliente*/
$queryTlf = "SELECT telefono FROM telefono_cliente where id_cliente=".$idCliente;
$resultTlf = mysql_query($queryTlf);
while($telefonoA=mysql_fetch_assoc($resultTlf)){
$todosTls=$todosTls.$telefonoA["telefono"];
}
mysql_free_result($resultTlf);

$tlf=str_split($todosTls,9); //Separar teléfono por teléfono

/* VALIDACIÓN DE CÓDIGO DE PROVINCIA */
$codlocalNum=str_split($tlf[0],2);
$numLocal=strstr($codlocalNum[0],$codLocal);
if($numLocal!==false)
{
$tlf[0]=$codlocalNum[1].$codlocalNum[2].$codlocalNum[3].$codlocalNum[4];
}

$codlocalNum=str_split($tlf[1],2);
$numLocal=strstr($codlocalNum[0],$codLocal);
if($numLocal!==false)
{
$tlf[1]=$codlocalNum[1].$codlocalNum[2].$codlocalNum[3].$codlocalNum[4];
}

$codlocalNum=str_split($tlf[2],2);
$numLocal=strstr($codlocalNum[0],$codLocal);
```

```
if($numLocal!==false)
{
$tlf[2]=$codlocalNum[1].$codlocalNum[2].$codlocalNum[3].$codlocalNum[4];
}

/*CREO EL ARCHIVO DE TEXTO CON EL CONTENIDO PARA EL ARCHIVO DE AUDIO */

$numCanal=$estado["num_canal"]; //asocio el número del canal por el que se va a realizar la llamada con el nombre del archivo de audio

$archivo=fopen("/etc/asterisk/outboundCall/archivoAudio$numCanal.txt","w+") or
die("Problemas en la creacion:");
fputs($archivo,"Estimado $nombre, Oroquil informa que adeuda la cantidad de $total dolares correspondientes a $numPagos meses. La fecha de pago limite es");
fclose($archivo);

/*CREO EL ARCHIVO DE AUDIO QUE SE REPRODUCIRÁ AL CLIENTE EN LA LLAMADA */

echo exec("text2wave /etc/asterisk/outboundCall/archivoAudio$numCanal.txt -F 8000 -o /var/lib/asterisk/sounds/outboundCall/archivoAudio$numCanal.wav");

$fechaU=strtotime($fecha);

/**** TERMINA OBTENCION DE DATOS ****/

/*LLAMA A 1ER NUMERO de Cliente*/
```

```
for ($i=0;$i<2;$i++) {
$fail1=$llamada1-
>realizarLlamada($socket,"$tlf[0]","60000","salientes","200","1","llamada1",$numCanal,$idCli
ente, $fechaU);

if($fail1==0)
{
$escuchoMsg=$llamada1->presionaCinco($socket);

if($escuchoMsg==1)
{
$i=2;
$sgteLlamada=0;
}
}
else
{
$i=2;
}
}

/*LLAMA A 2do NUMERO de Cliente*/
for ($j=0;$j<2 && $sgteLlamada==1;$j++) {

$fail2=$llamada2-
>realizarLlamada($socket,"$tlf[1]","60000","salientes","200","1","llamada2",$numCanal,$idCli
ente, $fechaU);
```

```
if($fail2==0)
{
$escuchoMsg=$llamada2->presionaCinco($socket);

if($escuchoMsg==1)
{
$j=2;
$sgteLlamada=0;
}
}
else
{
$j=2;
}
}

/*LLAMA A 3ER NUMERO de Cliente*/
for ($k=0;$k<2 && $sgteLlamada==1;$k++) {

$fail3=$llamada3-
>realizarLlamada($socket,"$tlf[2]","60000","salientes","200","1","llamada3",$numCanal,$idCli
ente, $fechaU);

if($fail3==0)
{
$escuchoMsg=$llamada3->presionaCinco($socket);
```

```
if($escuchoMsg==1)
{
$k=2;
}
if($i==1 && $escuchoMsg==0)
{
$queryStdLlamada = "UPDATE cliente SET estado_llamada1=1 where
no_cuenta=$idCliente";
$resultStdLlamada4 = mysql_query($queryStdLlamada);
mysql_free_result($resultStdLlamada4);
}
}
else
{
$queryStdLlamada = "UPDATE cliente SET estado_llamada1=1 where
no_cuenta=$idCliente";
$resultStdLlamada3 = mysql_query($queryStdLlamada);
mysql_free_result($resultStdLlamada3);
$k=2;
}
}
}
else{$numElse++;}
}
if($numReg==$numElse){echo "Se ha llamado por primera vez a todos los clientes";}
```

```
/**** DETIENE EL SCRIPT CUANDO SON MÁS DE LAS 8PM ****/
$fecha=date("H:i:s");
if(strcmp($fecha,"20:00:00")>0){

fclose($socket);  //termina conexion a manager

mysql_free_result($resultMonto); //termina conexion a base
mysql_close($link);

break; //rompe el while
}

/**** TERMINA CONEXION A MANAGER ****/
fclose($socket);

/**** TERMINA CONEXION A BASE ****/
mysql_free_result($resultMonto);
mysql_close($link);
?>
```

ANEXO B: Script de la clase Llamada

```
<?php
class llamada {
var $response;

public function
realizarLlamada($socket,$channel,$timeout,$context,$exten,$priority,$actionId,$numCanal,$i
dCliente,$fecha){

$fail=0;

fputs($socket, "Action: Originate\r\n");
fputs($socket, "Channel: DAHDI/$numCanal/$channel\r\n");
fputs($socket, "Timeout: $timeout\r\n");
fputs($socket, "Context: $context\r\n");
fputs($socket, "Exten: $exten\r\n");
fputs($socket, "Priority: $priority\r\n");
fputs($socket, "Variable: IDCLIENTE=$idCliente\r\n");
fputs($socket, "Variable: FECHA=$fecha\r\n");
fputs($socket, "Variable: AUDIO=$numCanal\r\n");
fputs($socket, "ActionId: $actionId\r\n\r\n");

do{
$response = fread($socket, 5120);
echo 'resp: '.$response;
//echo $response;
```

```
if(strstr($response, "Error")!==false) //cuando se corta la llamada
{
$fail=1;
}

}while (strstr($response, "$actionId")==false);

if(strstr($response, "Error")!==false) //cuando se deja timbrar
{
$fail=1;
}

return $fail;
}

public function presionaCinco($socket)
{
$escuchoMsg=0;

do{
$response = fread($socket, 5120);
echo 'resp1: '.$response;
//echo $response;
if(strstr($response, "Cause: 98")!==false) //presionó 5
{
$escuchoMsg=1;
}
```

```
}while (strstr($response, "Cause-txt")==false);


if(strstr($response, "Cause: 98")!==false) //presionó 5

{

$escuchoMsg=1;

}


return $escuchoMsg;

}

}

?>
```

ANEXO C: Aplicación Web

```
<?php
echo "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"
\"http://www.w3.org/TR/html4/loose.dtd\">";
echo "<html>";
echo "<head>";
echo "<script src='mootools-1.2.5-core-nc.js' type='text/javascript'></script>";
echo "<script>
window.addEvent('domready', function(){
$('primeraLlamada').addEvent('click', function(evento){

var canal1 = new Request({
method: 'get',
url: 'ejecutaAmic1.php',
}).send();

var canal2 = new Request({
method: 'get',
url: 'ejecutaAmic2.php',
}).send();

var canal3 = new Request({
method: 'get',
url: 'ejecutaAmic3.php',
}).send();
```

```
var canal4 = new Request({
method: 'get',
url: 'ejecutaAmic4.php',
}).send();



});

$('segundaLlamada').addEvent('click', function(evento){

var canal21 = new Request({
method: 'get',
url: 'ejecutaAmi2c1.php',
}).send();

var canal22 = new Request({
method: 'get',
url: 'ejecutaAmi2c2.php',
}).send();

var canal23 = new Request({
method: 'get',
url: 'ejecutaAmi2c3.php',
}).send();

var canal24 = new Request({
method: 'get',
```

```
url: 'ejecutaAmi2c4.php',
}).send();



});
});
</script>";
echo "</head>";
echo "<body>";
echo "<div style='width:750px; height:550px; border:double black 1px; margin:auto;'>";
echo "<div style='width: 700px; margin: auto; border:solid white 1px;'>";
echo "<h1 align='center'>Administrador de Generador de Llamadas
Autom&aacute;ticas</h1>";
echo "<h2>Instrucciones de Uso: </h2>";
echo "<ol>";
echo "<li><p align='justify'>Las llamadas se realizarán desde el momento en que se
presionen los botones de <i>Generar Llamada</i>, hasta las 20H00.</p></li>";
echo "<li><p align='justify'>El primer d&iacute;a del mes se deben empezar a realizar las
llamadas a los clientes, presionando el bot&oacute;n <b>Generar Primera Llamada</b>.
Este Proceso durar&aacute; aproximadamente 5 d&iacute;as, dependiendo del
n&uacute;mero de clientes presentes en la base de datos.</p></li>";
echo "<li><p align='justify'>A los clientes que no contestaron en el primer grupo de llamadas
se los deber&aacute; volver a llamar, para esto se debe presionar el botón <b>Generar
Segunda Llamada</b>.</p></li>";
echo "<li><p align='justify'>Cuando se ha terminado todo el proceso de las llamadas, se
debe generar un reporte de los clientes que no contestaron las llamadas. Para esto se debe
presionar el botón <b>Generar Informe</b>.</p></li>";
```

```
echo "</ol>";
echo "</div>";
echo "<div style='width:600px;border:solid white 1px; margin:auto;'>";
echo "<button id='primeraLlamada' style='width: 200px; height:50px;'>Generar Primera
Llamada</button>";
echo "<button id='segundaLlamada' style='width: 200px; height:50px;'>Generar Segunda
Llamada</button>";
echo "<button style='width: 200px; height:50px;'
onclick=\"javascript:location.href='http://localhost/empresaCobranza/pdf/generaInforme.php'\"
>Generar Informe</button>";
echo "</div>";
echo "</div>";
echo "</body>";
echo "</html>";
?>
```

ANEXO D: Script que Genera informe de llamadas

```
#!/usr/bin/php -q
<?php
ini_set("default_charset", "utf-8");

//Crear el documento PDF
require_once('class.ezpdf.php');
$pdf =& new Cezpdf('a4');$pdf->selectFont('../fonts/courier.afm');
$pdf->ezSetCmMargins(1,1,1.5,1.5);

//Obtenemos los registros desde MySQL
$conexion = mysql_connect("localhost", "admin", "pass_admin");
mysql_select_db("base_clientes", $conexion);

$queryMonto = "select cliente.nombre, SUM(monto.valor), cliente.fecha_pago,
cliente.fecha_llamada, cliente.estado_llamada, cliente.estado_llamada1 FROM monto,
cliente WHERE monto.id_cliente = cliente.no_cuenta GROUP BY monto.id_cliente";
$resultMonto = mysql_query($queryMonto);

$ixx = 0;
$jxx = 0;
while($monto=mysql_fetch_assoc($resultMonto)){
//echo "entra while\n";
if($monto["estado_llamada"]==0 && $monto["estado_llamada1"]==1)
{
```

```
$ixx = $ixx+1;
$dataNo[] = array_merge($monto, array('num'=>$ixx));
}
else
{
$jxx = $jxx+1;
$data[] = array_merge($monto, array('num'=>$jxx));
}

}

$titles= array(
"num"=>'<b>Num</b>',
"nombre"=>'<b>Nombre</b>',
"SUM(monto.valor)"=>'<b>Valor</b>',
"fecha_llamada"=>'<b>Fecha en que se realizó la llamada</b>',
"fecha_pago"=>'<b>Fecha máxima de pago</b>'
);

$options = array(
"shadeCol"=>array(0.9,0.9,0.9),
"xOrientation"=>'center',
"width"=>500
);

$txttit = "<b>EMPRESA DE GESTION DE COBRANZA</b>\n";
```

```
$txttit.= "REPORTE DE LLAMADAS\n\n";

$txttit.= ".:: LLAMADAS NO CONTESTADAS ::.\n";

$pdf->ezText($txttit, 12);

$pdf->ezTable($dataNo, $titles, '', $options);

$pdf->ezText("\n\n\n", 10);

$txttit2= ".:: LLAMADAS CONTESTADAS ::.\n";

$pdf->ezText($txttit2, 12);

$pdf->ezTable($data, $titles, '', $options);

$pdf->ezText("\n\n\n", 10);

$pdf->ezText("<b>Fecha:</b> ".date("d/m/Y"), 10);

$pdf->ezText("<b>Hora:</b> ".date("H:i:s")."\n\n", 10);

$pdf->ezStream();

?>
```