CAPÍTULO 2: ESTUDIO DEL SOFTWARE PBX VOIP: ASTERISK Y FREESWITCH

En este capítulo descubriremos qué es exactamente una PBX y cuál es su funcionamiento. Estudiaremos la centralita Asterisk, probablemente la más importante, y una de sus grandes competidoras, FreeSwitch. Presentaré brevemente las funcionalidades que ofrecen, comentaré su arquitectura y describiré algunos de los escenarios donde pueden actuar. Para terminar, realizaré una comparativa de ambas e identificaré las dificultades con las que me he encontrado tras su uso, así como la lectura y análisis de documentos críticos.

1. ¿Qué es una PBX?

Un PBX o PABX (**Private Branch Exchange** y **Private Automatic Branch Exchange** respectivamente) puede considerarse como cualquier central telefónica conectada directamente a la red pública de teléfono por medio de líneas troncales para gestionar, además de las llamadas internas, las entrantes y/o salientes con autonomía sobre cualquier otra central telefónica. Este dispositivo generalmente pertenece a la empresa que lo tiene instalado y no a la compañía telefónica, de aquí el adjetivo **privado** de su denominación. Un PBX se refiere al dispositivo que actúa como un ramificación de la red primaria pública de teléfono, por lo que los usuarios no se comunican con el exterior mediante líneas telefónicas convencionales, sino que al estar el PBX directamente conectado a la RTB (Red Telefónica Pública), será esta misma la que enrute la llamada hasta su destino final mediante enlaces unificados de transporte de voz llamados **líneas troncales**. En otras palabras, **los usuarios de un PBX no tienen asociada ninguna central de teléfono pública**, ya que es el mismo PBX que actúa como tal, análogo a una central pública que da cobertura a todo un sector, pero una PBX lo ofrece a las instalaciones de una compañía generalmente. No tienen límite en cuanto al número de estaciones que pueden servir, pero el precio aumenta según el número de estaciones. Los sistemas PBX son más sofisticados que los equipos **multilínea** o los **sistemas híbridos**, pero también son más costosos. La capacidad de un PBX no se determina por líneas, sino por los puertos de que dispone.

2. Plataforma Asterisk

En esta segunda parte, presentaremos la plataforma de telefonía y centralita telefónica Asterisk [5], describiendo para ello su arquitectura interna, algunos escenarios donde puede actuar así como los dispositivos que tiene que disponer un usuario final para beneficiarse de sus servicios.

2.1.¿Qué es Asterisk?

Asterisk es tanto una plataforma de telefonía como una centralita software (PBX) de código abierto [6]. Como cualquier centralita PBX permite interconectar teléfonos y conectar dichos teléfonos a la red telefónica pública. Su nombre viene del símbolo asterisco (*) en inglés. El creador original de esta centralita es **Mark Spencer** de la compañía **Digium**, que sigue siendo el principal desarrollador de las versiones estables. Al ser de código libre, existen multitud de desarrolladores que han aportado funciones y nuevas aplicaciones. Originalmente fue creada para sistemas **Linux** pero hoy en día funciona también en sistemas OpenBSD, FreeBSD, Mac OS X, Solaris Sun y Windows aunque Linux sigue siendo la que más soporte presenta. El paquete básico de Asterisk incluye muchas características que antes sólo estaban disponibles en caros sistemas propietarios como **creación de extensiones, envío de mensajes de voz a e-mail, llamadas en conferencia, menús de voz interactivos y distribución automática de llamadas**. Además se pueden **crear nuevas funcionalidades** mediante el propio lenguaje de Asterisk o módulos escritos en C o mediante scripts AGI escritos en Perl o en otros lenguajes. Asterisk soporta numerosos protocolos de VoIP como SIP y H.323 y puede operar con muchos teléfonos SIP, actuando como **servidor de registro**, o como "**gateway**" entre teléfonos IP y la red telefónica convencional. Al soportar una mezcla de la telefonía tradicional y los servicios de VoIP, Asterisk permite a los desarrolladores construir nuevos sistemas telefónicos de forma eficiente o migrar de forma gradual los sistemas existentes a las nuevas tecnologías

2.2.Arquitectura de Asterisk

La figura a continuación nos muestra los diferentes subsistemas que forman Asterisk.

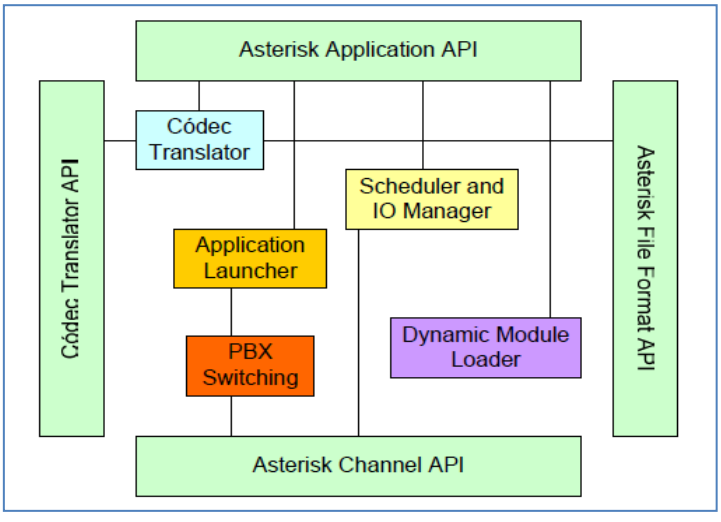

Figura 12: Subsistemas de Asterisk
(Fuente: Business Interactive. Web-Based Training VoIP Basics)

Cada uno de ellos realiza una función diferente.

- **Dinamic Module Loader**: cuando iniciamos Asterisk, es el encargado de cargar e inicializar los drivers necesarios.
- **PBX Switching**: es el encargado de aceptar y conectar las llamadas que recibe por las interfaces. Actúa según lo definido en el plan de numeración.
- **Application Launcher**: el PBX Switching utiliza este subsistema para lanzar las aplicaciones que sean necesarias como por ejemplo hacer sonar un teléfono, hacer saltar el buzón de voz de un usuario, etc.
- **Codec Translator**: codifica y decodifica los formatos de audio en el caso de que dos canales usen un códec diferente.
- **Scheduler and I/O Manager**: es el encargado de organizar las tareas de bajo nivel y de gestionar la entrada/salida para conseguir un óptimo rendimiento en todo tipo de situaciones de carga de trabajo.

Hay cuatro API's (Application Programming Interface) definidas para proporcionar funcionalidad de forma independiente al tipo de hardware y al protocolo que se usen.

- **Channel API**: esta API provee mecanismos para manejar el tipo de conexión que se está utilizando en una llamada (VoIP, ISDN, etc.).
- **Application API**: proporciona la funcionalidad para ejecutar aplicaciones tales como el buzón de voz, directorio de llamadas, etc.
- **Codec Translator API**: contiene los módulos de diferentes códecs para dar soporte a los diferentes formatos de audio y vídeo.
- **File Format API**: proporciona la funcionalidad necesaria para la lectura y escritura de diferentes formatos de archivo que se almacenan en el sistema.

2.2.1. Concepto de Canales en Asterisk

El concepto de canal en Asterisk es de suma importancia. Asterisk utiliza un canal para comunicarse con los clientes que utilicen un protocolo o una tecnología determinada.

Así, podemos identificar entre otros los siguientes canales:

- **Canal SIP**: será el canal que se utilice para las comunicaciones que se envíen o reciban de clientes SIP.
- **Canal H.323**: es el canal que se usará para las comunicaciones de los clientes que utilicen el protocolo H.323.

- **Canal IAX2**: lo mismo pero para los clientes que utilicen el protocolo IAX2.
- **Canal Dahdi**: es el canal que utiliza Asterisk para comunicarse con las líneas provenientes de la RTB tanto analógicas como digitales.

A modo de ilustración podemos observar la Figura 37, donde los canales están representados a modo de "tuberías" que tiene Asterisk por donde entran y salen las comunicaciones de los clientes pertenecientes a determinados protocolos o tecnologías.

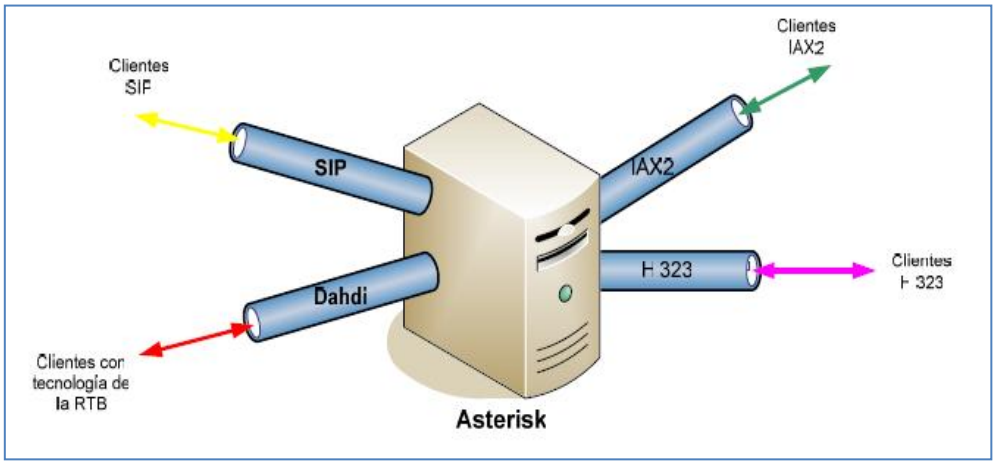

Figura 13: Canales en Asterisk
(Fuente: Business Interactive. Web-Based Training VoIP Basics)

Como norma general, Asterisk tendrá un archivo de configuración por cada canal. En ellos definiremos aspectos generales del protocolo o tecnología en cuestión así como información de los clientes que lo utilizarán.

2.2.2. Ejemplo de llamada

A continuación podemos observar cómo se realiza una llamada entre un cliente SIP y un puerto PRI. Según el concepto de canales anteriormente explicado, la invitación del cliente SIP es gestionada por el *monitor thread,* accediendo a través de un canal SIP y generando un nuevo hilo para que sea procesado por el *PBX Core*, el cual encamina la petición hacia el canal **Dahdi**, que es el encargado de comunicar con las líneas de la RTB.

La estructura de la llamada es definida en *struct ast_chan.* El hilo generado por la entrada de la llamada es reutilizado para la salida, pudiéndose asignar la memoria de la pila de proceso directamente.

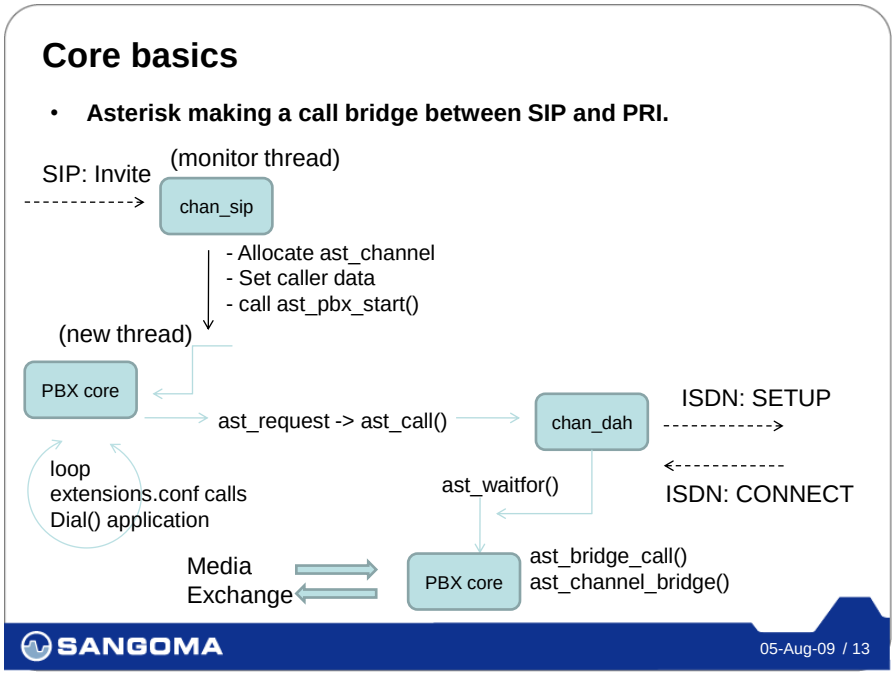

Figura 14: Ejemplo de llamada en Asterisk
(Fuente: http://www.sangoma.com)

2.3.Aplicaciones

Asterisk ha sido creado para funcionar como PBX, pero son muchas las aplicaciones que pueden construirse con esta plataforma:

- **Buzón de voz**: sistema de contestador automático personalizado por usuario. Se integra con el sistema de directorio (LDAP) y con el email.
    - Protección por contraseña.
    - Saludos personalizados por defecto.
    - Múltiples carpetas por defecto.
    - Interfaz web para comprobación de Buzón de Voz.
    - Notificación por e-mail del buzón de voz con archivo de audio adjunto.
    - Reenvío de buzón de voz.
    - Indicador visual de mensaje en espera (MWI).
    - Mensaje de tono de marcado de espera.
- **Gateway**: conecta equipamiento de telefonía a modernos sistemas de voz sobre IP y servicios.

- **Sistema de conferencias**: sistema que permite la conexión remota de diferentes usuarios que quieren mantener una reunión virtual y suministra la correcta gestión y control de los usuarios que se incorporan a ella.
    - No hay temporizadores.
    - No tiene características avanzadas DTMF activadas.
    - Reproduce sonidos para la programación de usuarios.
- **Voz de Respuesta interactiva (IVR):** Operadora Automática. Sistema automatizado de respuesta que permite redirigir las llamadas entrantes en función de las opciones seleccionadas por el llamante.
- **Sistema automático de llamadas (ACD):** Sistema Automático de Distribución de Llamadas entrantes. Pensado para Centros de Llamadas para atención comercial o soporte técnico.
    - Habilidades de enrutamiento.
    - Priorización de colas.
    - Instalación de IVR con GUI Dialplan Builder.
    - Interfaz de servicios web para bases de datos.
    - Múltiples DID/DNIS para servicios entrantes.
    - Constructor de Script con aplicaciones web embebidas.
    - Recapitulación de códigos personalizable y DND.
    - Transferencias (en vivo, asistidas y de consulta).
    - Temporización de la programación del día.

2.4.Configuración Asterisk

Después de instalar Asterisk para su configuración es necesario editar varios ficheros[7]. Todos ellos se encuentran en el directorio */etc/asterisk* y tienen extensión *".conf"*. En él se indican las rutas donde Asterisk debe ir a buscar o grabar diferentes archivos: los de configuración, los de audio, los de depuración, etc. En este apartado comentaremos las configuraciones más importantes para realizar la comunicación por voz IP, es decir, el **Plan de Marcación**, canal **Dahdi** y el **canal SIP.**

2.4.1. Plan de Marcación

El plan de marcación ("dialplan" en inglés) es sin duda el corazón de Asterisk. En él se especifica qué hacer con las llamadas que llegan y a dónde dirigir las que salen. De forma muy general, podemos decir que éste consiste una serie de pasos que Asterisk debe seguir cuando llega una llamada. Para configurarlo es necesario editar el archivo *extensions.conf* utilizando para ello una sintaxis apropiada.

Para configurar el plan de marcación, es imprescindible entender los siguientes conceptos clave: **contextos**, **extensiones**, **prioridades** y **aplicaciones.**

2.4.1.1. Contextos

El dialplan o lógica de comportamiento de Asterisk se divide en uno o varios contextos. Un contexto es una **colección de extensiones**. Es una forma de diferenciación. Existen para poder diferenciar el 'lugar' donde se encuentra una llamada, para:

- Aplicar políticas de seguridad: Asterisk no se comporta igual cuando llama un usuario y marca el 1 y cuando un usuario local marca el mismo 1.
- Menús y submenús diferenciados.

2.4.1.2. Extensiones

En telefonía tradicional, las extensiones se asocian con teléfonos, interfaces o menús. En Asterisk, una extensión es una **lista de comandos a ejecutar**.

A las extensiones se accede cuando:

- Se recibe una llamada entrante por un canal dado.
- El usuario que ha llamado marca la extensión.
- Se ejecuta un salto de extensiones desde el dialplan de Asterisk.

2.4.1.3. Prioridades

Una extensión puede tener varias instrucciones o pasos. La prioridad indica el orden que se va a seguir Asterisk al ejecutar todos los pasos. Si tenemos una extensión con muchos pasos y queremos introducir uno nuevo al principio utilizamos **prioridades no numeradas**. En vez de utilizar una secuencia de números empezando por 1, utilizamos la letra "n" (que indica next). Asterisk empezará por la prioridad 1 (la cual sí es necesaria ponerla), y cuando llegue al paso 2 y se encuentre una n, sumará uno a la prioridad

anterior. Esto hace mucho más fácil el introducir cambios en las extensiones.

2.4.1.4. Aplicaciones

Una aplicación es una determinada acción que se realiza en un canal de comunicación. Algunas de las aplicaciones más básicas de Asterisk se describen a continuación:

- **Answer ()**: si un canal está sonando, esta aplicación nos sirve para responder
- **Hangup ()**: hacemos terminar la conversación.
- **Playback ()**: sirve para reproducir por el canal de comunicación un fichero de audio que se encuentra almacenado en el sistema.
- **Wait ()**: nos sirve para esperar un determinado número de segundos antes de realizar la siguiente acción.

2.4.2. Canal Dahdi

Dahdi (Digium Asterisk Hardware Device Interface) es, a partir de la versión 1.4.22 de Asterisk, el **nuevo nombre para el canal Zaptel**. Éste nos permite la comunicación con usuarios que están en redes que utilizan tecnologías analógicas o digitales como la RDSI o PPP.

Con respecto a las versiones anteriores a la 1.4.22 de Asterisk, la configuración pasa de hacerse en un solo archivo *zaptel.conf* a hacerse en dos:

/etc/dahdi/system.conf y */etc/asterisk/chan_dahdi.conf*.

En el primero de ellos, */etc/dahdi/system.conf*, se configuran los datos relativos a las interfaces físicas entre Asterisk y el mundo exterior. Consta de muchos parámetros ya que contempla numerosas formas o tecnologías de conexión: analógicas, digitales, de radio, etc.

En el segundo, */etc/asterisk/chan_dahdi.conf*, se encuentran numerosos parámetros relativos a la configuración del canal lógico Dahdi. Podremos configurar aspectos como el contexto en el que entrará la llamada, su número de buzón de voz, si acepta o no transferencia de llamadas, etc. Además podremos configurar el método usado para la detección de colgado de un canal, regular la ganancia en el audio recibido o transmitido por Asterisk, agrupar canales (agrupar un número de líneas para que a la hora de utilizarlas para una comunicación podamos referirnos al grupo y no a una sola línea), etc.

2.4.2.1. Realizar llamadas con el canal Dahdi

A pesar de que todavía no tenemos configurado el canal SIP, con lo que no disponemos de clientes para realizar llamadas, vamos a ver qué necesitamos para realizar llamadas a móviles desde la empresa.

Lo primero de todo, es disponer de una aplicación que nos permita efectuar una llamada por un canal. Dicha aplicación es ***Dial (),*** la cual tiene el siguiente formato:

Dial (canal/identificador, tiempo_límite, opciones)

Para tener una extensión a través de la cual llamemos a los teléfonos móviles, debemos editar el fichero *extensions.conf* como mencionamos anteriormente.

Es importante señalar que cuando una llamada entra por el canal Dahdi por una línea, esta lo hará al contexto que hayamos definido para ella y siempre con la extensión "s" (del inglés "start"), que es la extensión por la que se empieza cuando entramos en un contexto sin que tengamos información sobre el identificador de llamada.

Para enviar una llamada de la extensión "s" a la extensión "NumeroN" utilizaríamos *Goto (),* la cual nos permite saltar a una determinada extensión, prioridad o contexto. Su formato es el siguiente:

Goto ([[contexto|] extensión|] prioridad)

Donde contexto y extensión son opcionales mientras que la prioridad es obligatorio indicarla.

2.4.3. Configuración del canal SIP

Para poder recibir llamadas por el canal SIP, es necesario configurarlo. Esta tarea se hace editando el archivo de configuración *sip.conf*. El fichero se divide en secciones las cuales empiezan con el nombre de la sección escrito entre corchetes:

[nombre sección]

La primera sección se llama "*general*" y en ella se definen las opciones generales del canal así como las opciones por defecto de todos los clientes contenidos en el fichero.

Las secciones siguientes comienzan con el nombre del cliente entre corchetes seguido de sus opciones.

2.4.3.1. Parámetros generales del canal SIP

Los parámetros más importantes del canal son:

- **port**: es el puerto que Asterisk utilizará para escuchar conexiones SIP entrantes. El valor por defecto establecido por los estándares es 5060.
- **bindaddr**: es la dirección IP que Asterisk utiliza para atender (o escuchar) conexiones SIP entrantes. El valor por defecto es 0.0.0.0, es decir, por defecto SIP escucha en todas las interfaces de red disponibles.
- **context**: es el contexto por defecto para las llamadas entrantes y se aplica cuando un cliente no tiene definido un contexto específico.
- **videosupport**: la opción *videosupport* nos permite, o no, habilitar el soporte de vídeo en SIP. Puede tomar los valores "no" o "yes". Siendo el "no", el valor por defecto.
- **language**: nos permite establecer el idioma por defecto para todos los clientes. Después es posible definir una lengua para cada uno de ellos.
- **allow**: aquí especificamos los codecs que queremos utilizar, y el orden de preferencia en el caso de que haya más de uno.

2.4.3.2. Definición de clientes SIP

Los clientes SIP deben ser **declarados y configurados** en el fichero antes de poder realizar o recibir llamadas a través del servidor Asterisk. Dentro de los múltiples parámetros que se pueden configurar, voy a comentar sólo los más básicos[1]:

- **type**: la opción *type* define la clase de conexión que tendrá el cliente. Hay tres tipos de clientes SIP:
  - **peer**: es un tipo de cliente que solo puede recibir llamadas.
  - **user**: es un tipo de cliente que solo puede hacer llamadas.
  - **friend**: es un tipo de cliente que es un peer y un user a la vez. Por lo tanto puede recibir y realizar llamadas.

[1] Todos los parámetros y su significado pueden consultarse en: Van Meggelen J, Madsen L, Smith J, (2007), Asterisk: The Future of Telephony, (2nd Edition), 0'Reilly, Appendix A: pp. 358 – 366.

- **secret**: con este parámetro establecemos la contraseña que el cliente usará para su autentificación.
- **host**: establece la dirección IP o el nombre de host del cliente. Puede tener el valor “dynamic” para cuando es un servidor DHCP quien asigna las direcciones IP a los clientes.
- **context**: es el contexto asociado al cliente.
- **callerid**: es el identificador del cliente.

2.5. Integración de Asterisk usando AGI y AMI

En muchas situaciones será necesario extender la funcionalidad de Asterisk usando **aplicaciones externas**. En centrales telefónicas convencionales, esto era normalmente hecho a través de una interfaz de integración de telefonía-computador conocida como CTI. El Asterisk como ya es construido sobre un computador, tiene diversas formas de integración y no está limitado apenas a una interfaz de CTI. En este capítulo veremos la interfaz de CTI de Asterisk conocida como AMI (Asterisk Manager Interface), y el uso de AGI (Asterisk Gateway Interface)

2.5.1. AMI (Asterisk Manager Interface)

El AMI (Asterisk Manager Interface) permite a un programa cliente conectarse a una instancia de Asterisk y emitir comandos o leer eventos del PBX sobre un flujo TCP/IP.

Existen ejemplos de programación de AMI en Java, Perl, Python, PHP, C, C# entre muchas otras. Es posible programar el AMI en cualquier lenguaje que suporte una interfaz de sockets o que simule Telnet.

2.5.1.1. Comportamiento del protocolo

- Antes de emitir cualquier comando para Asterisk, se debe establecer una sesión de control.
- La primera línea de un paquete tendrá una llave “Action” cuando es enviada a un cliente.
- La primera línea de un paquete tendrá una llave “Response” o “Event” cuando es enviada a partir de Asterisk.
- Paquetes que pueden ser transmitidos en cualquier dirección después de la autentificación.

A la vista de los procedimientos mencionados aclaramos la función de los distintos paquetes:

- **Action**: un paquete enviado por un cliente conectado al Asterisk solicitando que una acción sea hecha. Existe un conjunto finito de acciones disponibles para el cliente, determinada por los módulos actualmente cargados en el Asterisk. Solo una acción será procesada por vez. El paquete contiene el nombre de la operación y sus parámetros.
- **Response**: es la respuesta enviada por el Asterisk a la última acción solicitada por el cliente.
- **Event**: datos pertenecientes a un evento generado de dentro del núcleo de Asterisk o por un módulo.

2.5.1.2. Configurando usuarios y permisos

Para acceder al AMI es necesario establecer una conexión TCP/IP escuchando una puerta TCP (normalmente 5038). Para esto es necesario configurar el archivo */etc/asterisk/manager.conf* y crear una cuenta de usuario. Además de la cuenta es necesario configurar un conjunto de permisos. Existe un conjunto finito de permisos: *read* para lectura, *write* para escritura o ambos.

Para hacer el login y autenticarse en AMI, se precisa enviar una acción de tipo *Login* con su nombre de usuario y contraseña.

2.5.2. Asterisk Manager Proxy

Asterisk no fue hecho para conectar un gran número de conexiones a la interfaz del Manager. Por ejemplo, si cada supervisor o agente de un call-center usa el AMI para *login* y *logout* de agentes, esto **puede tornar al Asterisk inestable**. Fue desarrollado un proxy llamado "***astmanproxy"*** para resolver este problema. Las principales ventajas de Astmanproxy son:

- Una conexión única y persistente en el Asterisk.
- Una interfaz TCP más segura (no root).
- Habilidad de filtrar entrada / salida.
- Menor número de conexiones y carga para el Asterisk.
- Múltiples medios de acceso (estándar, http, XML, csv).
- Soporte para SSL.

- Conexión a múltiples servidores Asterisk.
- Formatos de I/O seleccionadles en una base por cliente.

2.5.3. Asterisk Gateway Interface

El Asterisk Gateway Interface, o AGI, proporciona una **interfaz estándar por la cual programas externos pueden controlar el dialplan de Asterisk**. Por lo general, los scripts AGI se utilizan para hacer lógica avanzada, comunicarse con bases de datos relacionales (como PostgreSQL o MySQL), y acceso a otros recursos externos. A través del control del dialplan de un script AGI externo, Asterisk permite realizar fácilmente las tareas que de otra manera serían difíciles o imposibles. Los programas externos pueden estar escritos en diferentes lenguajes de programación como C, C#, Bourne Shell, PHP, Ruby, Python, Perl, Java,…

AGI es llamado a partir del plan de discado definido en *extensions.conf*.

El programa debe cumplir:

- Ser ejecutable (mediante un intérprete es válido).
- Localizado por defecto en */var/lib/asterisk/agi-bin.*
- Asociado con alguna extensión en el Dial Plan

En cuanto a la comunicación, utiliza los canales estándar típicos, ***stdin,tdout y stderr***, correspondientes con la entrada estándar (para obtener una respuesta de Asterisk), salida estándar (para enviar un comando a Asterisk) y el canal para los errores respectivamente.

Existen tres tipos de AGIs:

- **AGI Normal**: llamado directamente en el servidor Asterisk.
- **FastAGI**: que llama al AGI en otro servidor a partir de una interfaz sockets.
- **EAGI:** da a la aplicación la posibilidad de acceder y controlar el canal de sonido más allá de la interacción con el plan de discado.
- **DEADAGI**: que da acceso a un mismo canal después de hangup ().

2.6.Asterisk en diferentes escenarios

Atendiendo a su función de centralita telefónica, Asterisk puede usarse en multitud de escenarios. Así, puede proveer servicios de telefonía a usuarios que se encuentran en una LAN y desean poder realizar llamadas entre ellos, o bien, yendo un poco más lejos, utilizar Internet para realizar llamadas a otros usuarios conectados a ésta o incluso, tener la capacidad de llamar a teléfonos convencionales conectados a la RTB. A continuación describiremos qué necesita Asterisk para poder dar servicio en cada uno de estos casos.

2.6.1. Llamadas internas en nuestra oficina

El escenario más simple que podemos imaginar para Asterisk, es proporcionar llamadas para los usuarios de una red la cual no está conectada con Internet. El ejemplo más claro de esto sería una red privada dentro de una empresa u oficina.

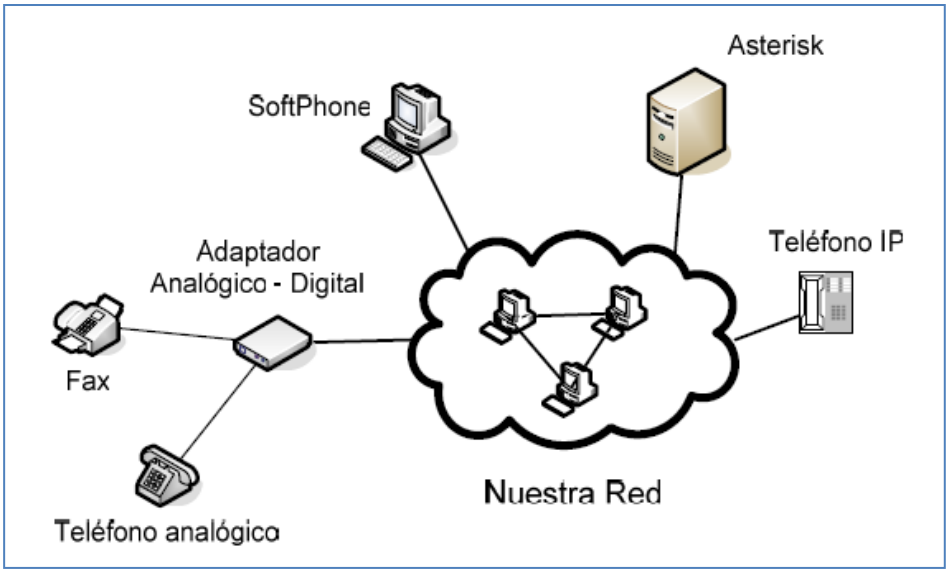

Figura 15: Asterisk en una red interna.

Como observamos en la Figura 4, sólo es necesario conectar el servidor Asterisk a la red IP mediante una NIC (Network Interface Card) y con la configuración pertinente los usuarios de la red podrán realizar llamadas entre ellos sin coste alguno.

2.6.2. Conectar nuestra oficina con Internet

En este segundo escenario, tenemos la misma red privada que antes pero conectada a Internet. Con este simple hecho, un empleado de nuestra oficina podrá **establecer llamadas con cualquier dispositivo VoIP que esté conectado a Internet sin coste alguno**.

Sólo habrá que tener en cuenta cuestiones tales como el ancho de banda necesario para soportar el aumento de tráfico, la seguridad, pero no habrá que incurrir en más gastos que la propia conexión a Internet. Imaginemos que una compañía con dos sedes, una en Sevilla y otra en Montreal por ejemplo, tienen ambas un sistema Asterisk. Si las dos están conectadas a Internet, se podrán realizar llamadas entre ambos sitios a través de la red.

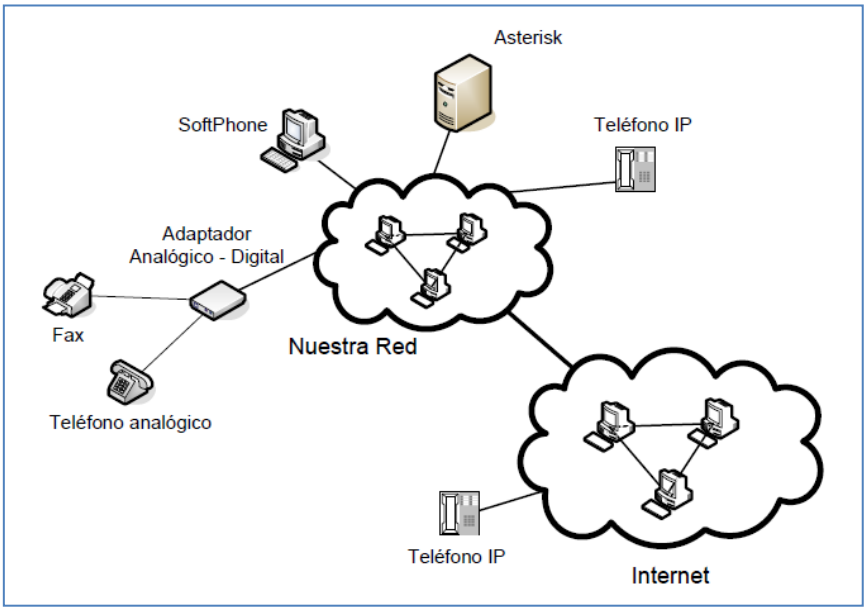

Figura 16: Asterisk conectado a Internet.

2.6.3. Conectar nuestra oficina con la RTB

En este último escenario, la red privada ya conectada a Internet se ha unido a la red telefónica pública. Para poder recibir llamadas de cualquier teléfono convencional de la RTB **necesitaremos disponer de líneas de teléfono proporcionadas por una compañía telefónica**.

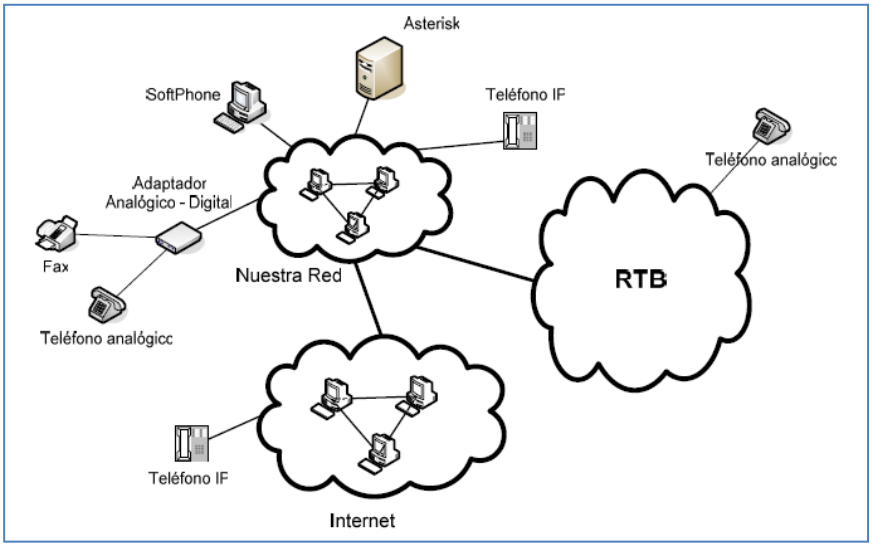

Figura 17: Asterisk conectado a la RTB

Las líneas telefónicas deberán llegar a Asterisk utilizando para ello una tarjeta conectada al bus PCI o un dispositivo externo denominado **pasarela VoIP**.

Ambos métodos proveen una interfaz entre la red IP de nuestra oficina con la red telefónica. A continuación describiremos las dos formas más usuales de conectarnos a la Red Telefónica Básica.

2.6.3.1. Conexión a la mediante una línea de teléfono básica

La forma más simple de conectarnos sería mediante una línea **POTS** (Plain Old Telephone Network) o línea de teléfono básica. Esto es la línea de teléfono que todos tenemos en casa. Esta línea representa un canal de transmisión y sólo se puede transmitir por él una llamada.

Una posible tarjeta para hacer esta conexión es comercializada por Digium, la misma empresa creadora de Asterisk. Se conecta al ordenador mediante el bus PCI y presenta una serie de puertos a los que le podemos conectar una combinación de líneas **FXS** (Foreign Exchange Station) y **FXO** (Foreign Exchange Office), según el modelo.

Recordemos que una interfaz FXS es la que maneja al teléfono proporcionándole la batería y el voltaje necesario para que suene. En nuestra casa, es ni más ni menos que la roseta en la pared donde conectamos el teléfono. Una interfaz FXO recibe este voltaje y hace sonar al teléfono. En este caso sería la clavija que introducimos en la roseta. En la Figura 48, podemos ver claramente la colocación de las estas interfaces en el camino de una llamada desde la central hasta un teléfono.

Figura 18: Interfaces FXS y FXO

Por lo tanto esta tarjeta nos permite conectar líneas telefónicas que vienen de la central a las interfaces FXS y teléfonos analógicos a las interfaces FXO. La segunda posibilidad es utilizar un **dispositivo externo** que haga de interfaz entre la red telefónica y la red IP. Es como si sacáramos la tarjeta antes comentada y la pusiésemos en un dispositivo aparte.

En la Figura 8 podemos ver la **pasarela VoIP Linksys SPA 3102**, que tiene un puerto FXO RJ11 para conectarlo a la roseta de teléfono, un puerto FXS para conectar un teléfono o fax analógico, un puerto Ethernet para conectarlo a la red y por último ofrece un puerto para conectarnos a Internet mediante DSL.

Figura 19: Modelo Linksys SPA 3102

2.6.3.2. Conexión mediante una línea RDSI

Otra forma muy común de conexión con la RTB es utilizar una línea RDSI (Red Digital de Servicios Integrados). Estas líneas utilizan el bucle local para brindar una comunicación digital entre el usuario y el operador de telefonía.

Para conectar Asterisk a la RTB mediante líneas de este tipo existen tarjetas que conectadas normalmente al bus PCI hacen de interfaz entre la red telefónica pública y nuestro sistema.

Según el número de líneas que queramos utilizar, disponemos de varios modelos de tarjetas de diferentes fabricantes. Si queremos usar un servicio BRI, las más conocidas son las tarjetas **Billion**, **Junghanns, Beronet** y las **Digium.**

Si queremos utilizar el servicio primario (PRI), es **Digium** el que brinda una mayor variedad de productos y además al estar integradas en Asterisk no necesitan ningún driver adicional.

3. Plataforma FreeSwitch

A continuación comentaremos la plataforma que se presenta como **gran alternativa a Asterisk**. FreeSwitch fue **inicialmente concebido como Soft-Switch**, pudiendo trabajar también como PBX. El creador y desarrollador, Anthony Minessale, fue uno de los desarrolladores de Asterisk. Después de añadir muchas nuevas e invaluables características a Asterisk concluyó que la aplicación tenía muchas limitaciones que podrían ser reparadas únicamente con una reescritura completa del software. Por ello, su creador ha desarrollado completamente esta nueva aplicación totalmente desde cero, usando solamente su experiencia previa en la colaboración con Asterisk.

3.1.¿Qué es FreeSwitch?

FreeSwitch es una **plataforma de telefonía de código abierto diseñada para facilitar la creación de productos de voz y chat, desde un soft-phone hasta un soft-switch** [8]. Puede ser utilizado como un motor de conmutación simple, una PBX, un *gateway* o un servidor para alojar aplicaciones IVR utilizando secuencias de comandos simples o XML para controlar el *callflow*.

Apoya las diversas tecnologías de comunicación como **Skype, SIP, H.323 y GoogleTalk**, lo que facilita la interrelación con otros sistemas PBX de código abierto como **sipXecs, Call Weaver, Bayona, YATE o Asterisk**.

FreeSwitch soporta muchas características avanzadas, tales como Presence, SIP/BLF/SLA, así como TCP TLS y SRTP. También se puede utilizar como un proxy transparente con y sin medios de comunicación en el camino para actuar como un SBC (Session Border Controller), proxy T.38 y otros protocolos extremo a extremo.

Soporta la banda ancha y estrecha de codecs por lo que es una solución ideal para

mejorar el legado de dispositivos para el futuro. Los canales de voz y el módulo de *bridge conference*, todos pueden funcionar a 8, 12, 16, 24, 32 o 48 KHz, y pueden cruzar canales de diferentes tasas.

Se construye y se ejecuta de forma nativa independiente en varios sistemas operativos incluyendo Windows, Mac OS X, Linux, BSD y Solaris, en ambas plataformas de 32 y 64 bits.

Los desarrolladores están muy implicados en el código abierto y han donado código y otros recursos para otros proyectos de telefonía, incluyendo OpenSER, sipXecs, The Asterisk Open Source PBX y Call Weaver.

3.2.Arquitectura FreeSwitch

FreeSwitch es una aplicación **modular** [9]. La capa de abstracción previene que los módulos no dependan entre sí. La meta es **asegurar que el núcleo no dependa de los módulos para ser iniciado y que los módulos no dependan de otros módulos para ser cargados**. En la siguiente figura podemos ver la arquitectura del núcleo de FreeSwitch:

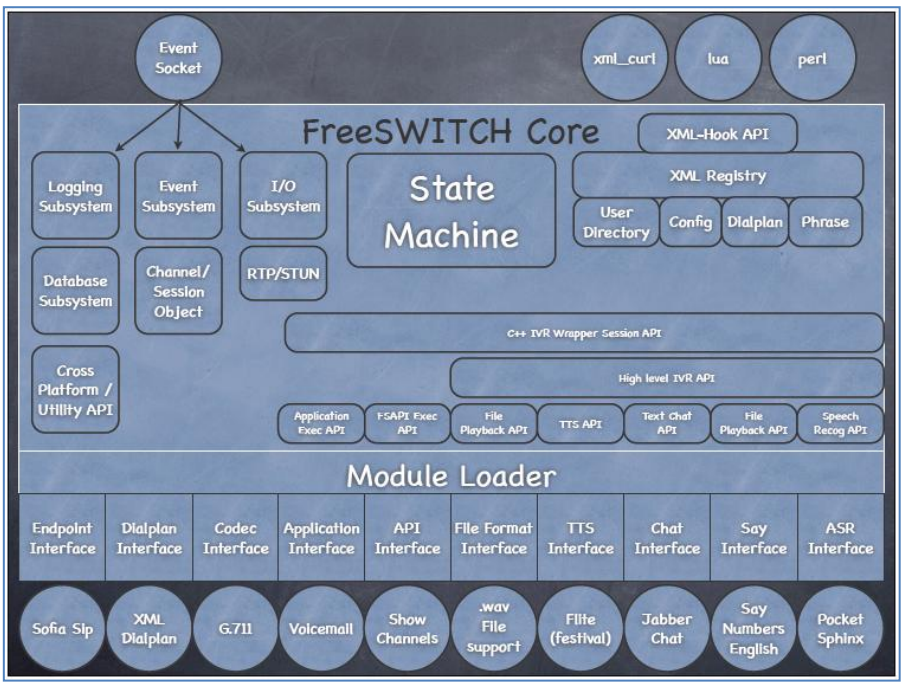

Figura 20: Arquitectura FreeSwitch
(Fuente: Michael Jerris. What is FreeSwitch)

Las características más importantes de esta arquitectura son:

- Modelo de hilos (operaciones paralelas).
  - Cada conexión tiene su propio hilo
  - Los sub-sistemas se ejecutan en hilos (o subprocesos) en segundo plano.
  - Manejadores de eventos ponen eventos en los hilos.
- Núcleo estable protegido.
  - Las estructuras de datos críticas son opacas.
  - El código complicado está todo en un mismo lugar.
  - Las rutinas complicadas consiguen la máxima reutilización.
  - El código de alto nivel no puede ser mal empleado.
  - Con cada nuevo nivel de API, la capacidad aumenta así como la complejidad disminuye.
- Módulos dinámicos para extender las funcionalidades.
  - La interfaz FSAPI permite la adición de comandos a los que se puede acceder de forma remota.
  - La Interfaz de Aplicación permite la creación de IVRs personalizadas y aplicaciones de enrutamiento de llamadas.
  - La Interfaz XML permite acceso dinámico en tiempo real para directorio de usuarios y configuración.
  - La interfaz Endpoints (Interfaz de puntos finales) conecta mensajería instantánea y teléfonos.
  - Formats, ASR/TTS, Say/Phrase Macros, amplían las capacidades de entrada.
- Sistema de Logger y eventos detallado.
  - Los eventos de desencadenan siempre que algo importante sucede.
  - Un registro detallado hace la depuración más sencilla.

- Las aplicaciones externas pueden enlazar a eventos y comprobar el estado del sistema.

- Enlaces para todo (Hooks into everything)
  - ¡Aprendamos a pensar en cuatro dimensiones!
  - Lenguajes de programación embebidos ayudan a simplificar la API aún más.
  - Basado en el mismo principio que el CGI/Web server.

3.2.1. Ejemplo de llamada

Podemos ver en la siguiente figura cómo se desarrolla en FreeSwitch una llamada entre un cliente SIP y el puerto PRI. Recibimos la invitación del cliente SIP y administrado por el *monitor_*thread será el *mod_sofia* el encargado de llamar al *switch_core_session_request* , establecer un perfil de llamada y de que se genere un nuevo hilo en la máquina de estado. Ésta enrutará y ejecutará el estado y pondrá en funcionamiento las funciones necesarias que, a través de *mod_openzap,* nos permitirán comunicarnos con la RTB.

La estructura de la llamada viene definida en *switch_core_session*. Cada sesión tiene su propio hilo de máquina de estado. Se asigna memoria usando fondo de memoria de sesión.

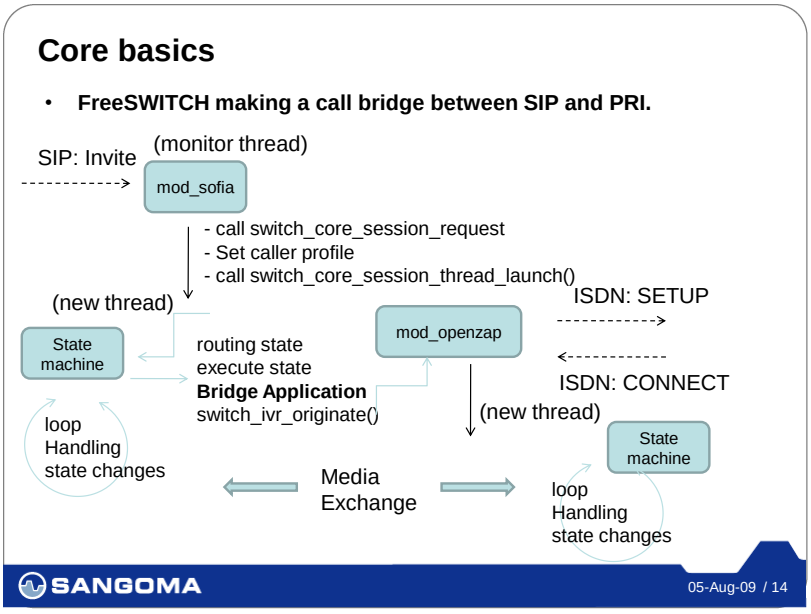

Figura 21: Ejemplo de llamada en FS
(Fuente: http://www.sangoma.com)

3.3.Aplicaciones

FreeSwitch fue inicialmente concebido para funcionar como soft-switch, pero son muchas más las aplicaciones que se pueden ejecutar con la plataforma:

- **Buzón de voz:** sistema de contestador automático personalizado por usuario.
    - Multiusuario. Configuración de empresa/compañía.
    - Temporización de saludos.
    - Marcado urgente de mensajes.
    - Entrega e-mail.
    - Reproducción y regrabado de mensajes antes de la entrega.
    - Las llaves son plantillas para que se puedan reorganizar según nuestras necesidades.
    - Soporte de devolución de llamada para buzón de voz entrante.
    - Podcast de buzón de voz (RSS).
    - Indicador de mensajes en espera (MWI).
- **Soporte para Colas** (via *mod_fifo*[2]): las llamadas entrantes pueden ser recibidas y organizadas en colas por medio de *mod_fifo.* Las llamadas que más tiempo lleven en la cola serán las primeras en ser atendidas (método FIFO, *first in, first out*).
- **Parking** (via *mod_fifo*): las llamadas pueden ser recibidas y "aparcadas" un cierto tiempo para ser procesadas en el momento indicado.
- **Conferencias:** sistema que permite la conexión remota de diferentes usuarios que quieren mantener una reunión virtual y suministra la correcta gestión y control de los usuarios que se incorporan a ella.
    - Software de Conferencia sin ningún requerimiento hardware.
    - Conferencias de banda ancha.
    - Múltiples conferencias bajo demanda o programadas con entrada/salida de anuncios.
    - Reproducción de archivos en conferencia o en usuario.
    - Relaciones.

[2] Para más información sobre el modo *mod_fifo* consultar http://wiki.freeswitch.org/wiki/Mod_fifo.

- Integración TTS.
- Transferencias.
- Llamadas salientes.
- Volumen, ganancia y nivel de energía por llamada.
- Puente a transición de conferencia.
- Marcación saliente multipartida.

- **Lector RSS**: Permite un índice de noticias de varios canales RSS a través del teléfono.
- **T.30 Audio Fax** (via *mod_fax*[3]): Envío y recepción de fax.

3.4.Configuración FreeSwitch

3.4.1. Introducción

FreeSwitch tiene un **diseño modular**. Este diseño permite que sea actualizado sin romper la funcionalidad o requerir gran esfuerzo de recodificación. Cada módulo ofrece funcionalidades específicas, y a menudo los comandos que están relacionados se agrupan en un módulo. Necesitaremos habilitar los módulos que deseemos en base a su función. De forma predeterminada viene con un conjunto de módulos cargados para habilitar la funcionalidad más básica.

Las siguientes secciones nos mostrarán cada paso que tenemos que dar para conseguir tener un sistema funcional. Antes de seguir tendremos que comprobar nuestro **firewall** y los **archivos de configuración**.

3.4.1.1. Firewall

Si vamos a enrutar el tráfico a través de nuestra red, es necesario asegurarse de que usted no tenemos un firewall bloqueando el tráfico en las rutas de acceso de red especificada. Los firewalls comúnmente se instalan en los puntos de salida, donde la red local se conecta a Internet. Esto no es siempre el caso en algunos entornos corporativos. Si usted no es responsable de la red, deberá ponerse en contacto con los responsables de ella.

[3] Para más información sobre el modo *mod_fax o* consultar http://wiki.freeswitch.org/wiki/Mod_fax.

La información que necesitaríamos para habilitar el tráfico podría incluir el número de puertos. Diversos protocolos utilizan puertos diferentes para diferentes cosas.

3.4.1.2. Archivos de configuración

Los datos de configuración en FreeSwitch se almacenan en formato XML. Estos archivos se encuentran en el directorio de configuración que tenemos bajo el directorio de instalación de FreeSwitch. Por defecto, en un sistema operativo tipo Unix que será */usr/local/freeSwitch/conf*. Hay etiquetas de marcado preprocesadas que le dicen al programa de análisis la configuración para realizar tareas, como puede ser la inclusión de otros archivos, llevándose a cabo antes de que el XML sea analizado.

Un pequeño resumen de estos archivos:

Archivos	Descripción
autoload_configs	Estos archivos de configuración son cargados automáticamente en FS. Contienen información de configuración de casi todos los módulos básicos.
dialplan	Este es el lugar donde configurar su dialplan. Hay algunos ejemplos de cómo configurarlo incluidos.
directory	El directorio contiene todos los usuarios que pueden registrar y utilizar FreeSwitch como su PBX.
jingle_profiles	Jingle es el mod que FS utiliza para manejar XMPP. Los perfiles le dicen a FS cómo utilizar el protocolo. Cada perfil tiene su puerto IP propio.
lang	Le dice a FS cómo decir algo en diferentes lenguas.
mrcp_profiles	MRCP se utiliza para permitir a FS utilizar reconocimiento de voz y TTS.
sip_profiles	Le dice a FS cómo hablar SIP. Cada perfil tiene su propio puerto.

Tabla 1: Archivos de configuración

3.4.2. Configurando FreeSwitch

Ahora vamos a explicar cómo configurar FreeSwitch a su gusto. La disposición que veremos no es en absoluto necesaria, pudiéndose minimizar a configurar un único archivo. EL archivo de configuración principal se llama *freeswitch.xml* y no es necesario cambiarlo, ya que carga todos los otros archivos de configuración[4].

3.4.2.1. vars.xml

Se utiliza para definir un par de variables que se utilizan en todo el sistema. En la configuración por defecto el archivo *vars.xml* se utiliza para definir algunas variables de preprocesador.

3.4.2.2. autoload_configs

Es un directorio donde reside una gran parte de la configuración FreeSwitch. Este directorio se encuentra en *$PREFIX/conf/autoload_configs*. El *freeswitch.xml* por defecto pre-procesa archivos XML correspondientes al *glob /autoload_configs/xml.*

3.4.2.2.1. modules.conf.xml

Le dice a FreeSwitch los módulos a cargar. Hay ciertos módulos necesarios para funcionar de manera que no se debería modificar este archivo a menos que sepa y se desee agregar o quitar un módulo específico.

3.4.2.2.2. sofia.conf.xml

Se utiliza para crear puntos finales SIP en FreeSwitch. Si está familiarizado con Asterisk, cada perfil de SIP en *mod_sofia* es similar a *chan_sip* en Asterisk, aunque **mucho más potente**.

El archivo *sofia.conf.xml* contiene una directiva "X-PRE-PROCESS" que incluye otros archivos XML (al menos, por defecto, *conf/sip_profiles/*) que definen uno o más perfiles SIP. Un perfil SIP es un **UA SIP** (agente de usuario o un punto final), que se comunica con otros terminales SIP.

[4] Ejemplos de los archivos de configuración pueden ser consultados en la wiki de FreeSwitch http://wiki.freeswitch.org/wiki/Getting_Started_Guide#Configuring_FreeSWITCH.

Un "User Agent" (UA) es una aplicación que se utiliza para ejecutar un protocolo de red determinado. Un “Sofía User Agent” es lo mismo pero el protocolo en este caso es SIP. En términos FreeSwitch, Agente de usuario = UA = perfil Sofía = perfil SIP.

3.4.2.3. ¿Qué hacen los perfiles SIP?

Los perfiles SIP en FreeSwitch a menudo pueden conducir a confusión. A diferencia de otros soft-switches (como Asterisk), **FreeSwitch le permite manejar los medios de comunicación (llamadas, vídeo, etc.) de forma diferente según el lugar donde el equipo está conectado a la red**. Esto ayuda con la seguridad, así como proporcionar mayor funcionalidad.

Los perfiles SIP permiten definir rutas de acceso a los dispositivos o a las compañías que pueden estar dentro o fuera de su red. Estas rutas pueden ser de muchos tipos diferentes, pero debe consistir en una combinación única de puerto y pares IP. Usted podría tener perfiles de SIP para la red interna, o varios perfiles para cada subred de su red interna, o incluso protocolos completamente diferentes como IPv6 como definiciones de perfil. Esto ayuda a FreeSwitch a identificar cómo localizar los diferentes tipos de llamadas cuando sea necesario, y también nos da flexibilidad para adaptar nuestro dialplan basándose en cuál es el camino de una llamada que creamos o recibimos.

3.4.2.3.1. Perfiles SIP

Los perfiles por defecto son **internos** y **externos**, cada uno tiene un propósito especial por lo que importante entender qué hacen.

3.4.2.3.1.1. Internos

Este perfil se refiere generalmente a los dispositivos que residen en la red interna. Estos archivos se encuentran en *$PREFIX/conf/sip_profiles/internal.xml.*

Estos dispositivos heredarán todas las opciones de configuración en el perfil interno que establezcamos. Esto es, típicamente nuestros teléfonos internos con extensiones. Por defecto, estos dispositivos deben utilizar la autenticación SIP.

El perfil interno SIP (antiguamente llamado "default") está configurado para escuchar en la dirección IP de la máquina (a menos que asignemos *$$(domain)* a algo en *vars.xml*) en el puerto 5060 (el

puerto SIP por defecto). El perfil SIP interior se necesita autentificar y no es adecuado para la configuración de trunks de proveedores o teléfonos externos, en la mayoría de los casos (por ejemplo, utiliza el perfil externo esto).

El perfil de interior se debe usar si se tiene la intención de manejar el registro de clientes SIP (es decir, un registro SIP).

Los usuarios autenticados de este perfil son normalmente configurados (a través de la *user_context*) para utilizar el dialplan en el contexto definido por defecto (*Config_default.xml*).

3.4.2.3.1.2. Externos

Este perfil se refiere generalmente a los dispositivos o gateways que residen fuera de nuestra red. El archivo se encuentra en *$PREFIX/conf/sip_profiles/external.xml.*

Generalmente es este el perfil donde se definen los trunks externos, carriers, etc. Una vez más, los dispositivos heredarán todas las opciones de configuración del perfil externo que establezcamos.

El perfil exterior (antes conocido como "outbound") maneja registros de salida a un proveedor SIP. El proveedor SIP nos envía las llamadas, y nosotros enviamos las llamadas a nuestro proveedor, a través del perfil externo.

Con el fin de garantizar FreeSwitch es conveniente enlazar su perfil de salida a un contexto dialplan que no sea "por defecto", donde la configuración por defecto es la que los usuarios autenticados quieran.

El puerto por defecto para las conexiones externas es 5080. Esto también significa que si se trata de conectar un teléfono de forma remota a su box FreeSwitch, debe establecer que se conecte en el puerto 5080, o cambiar los puertos de todo. Esto confunde a mucha gente. A modo de ejemplo, en X-Lite, si se conecta a FreeSwitch desde el exterior, hay que configurar nuestro servidor como *my.server.com: 5080*.

El dialplan para este perfil (por defecto) se define en el contexto público (*Config_external.xml).*

3.4.2.4. Dialplan

El dialplan de FreeSwitch no es una sola entidad. Tenemos la opción de ejecutar distintos subsistemas dialplan de forma nativa. Todos estos no son trasladados al mismo *backend* como otros sistemas que pueden ser empleados. Por el contrario son únicos, son métodos independientes a través de los cual se puede acceder a la información.

A diferencia otros switches, el dialplan no está diseñado para ser un lenguaje de secuencias de comandos que se pone sobre un montón de lógica. El dialplan, simplemente está diseñado para tomar una petición de llamada y decidir dónde se debe enviar y reenviar una aplicación. Por ejemplo, puede enrutar una llamada a la *bridge_Application ()*, y la aplicación permitirá hacer surgir un nuevo canal, y luego conectar los dos canales; podrá ir a la *Conference Application*, o cualquier otra aplicación registrada en el sistema FreeSwitch. Algunas de las aplicaciones más comunes que se deseen utilizar, tales como el *bridge*, se pueden encontrar en la documentación del módulo *dptools*.

El diseño para permitir múltiples módulos de procesado de *dialplan*, así como llamadas de enrutamiento a las aplicaciones que hacen todo el trabajo duro, le da la flexibilidad para hacer lo que necesites de la forma en que lo necesites para trabajar. No te obligan a adaptar su infraestructura en todo FreeSwitch, pero permite a FreeSwitch más facilidad de acoplamiento con la infraestructura existente.

Es importante saber que FreeSwitch posee múltiples contextos para evitar la exposición de extensiones internas a todo el mundo. Así, por defecto, los contextos son llamados "Public" y "Default" (pero estos nombres son arbitrarios y se utilizan en la configuración por defecto que viene con FreeSwitch). Todo en el contexto público está disponible para todos, mientras que en el contexto "default" sólo está disponible para los usuarios que se han registrado en FreeSwitch. Se puede definir en los perfiles de usuario (en el directorio) a qué contexto perteneces, y se puede definir el contexto público en la configuración de Sofía.

Algunos **módulos** importantes del dialplan son:

- **Dialplan XML:** el dialplan XML es el dialplan predeterminado que se utiliza. Esto crea un dialplan flexible, y permite para una tercera parte

de las herramientas ser creadas de forma y sencilla, ya que hay una gran cantidad de software que puede manipular dialplans. XML es también fácil y rápidamente modificado manualmente sin necesidad de herramientas especiales, como un editor de texto. El dialplan XML permite expresiones regulares en Perl, que le da una gran flexibilidad en un formato estandarizado.

- **FreeSwitch Dialplan Directory**: el Directorio dialplan permite a las consultas LDAP decidir dónde enrutar las llamadas.

- **Diaplan YAML:** YAML se llevó a cabo sólo hasta cierto punto y como se redactó el presente, el equipo central decidió no seguir aplicándolo. Por lo tanto seguir con XML.

- **Mod_diaplan_asterisk:** una vez cargado en *modules.conf.xml* puede ser utilizado en nuestro perfil de *Sofia* añadiendo "asterisk" como parámetro en el dialplan. También puede transferirle llamadas mediante la especificación de parámetros del dialplan en la transferencia o mediante aplicaciones *execute_extension*. No es del todo igual al dialplan real de Asterisk, pero al menos guarda una estrecha familiaridad. No puede incluir contextos en cualquiera de los otros contextos de FreeSwitch, porque **en FreeSwitch toda la lógica del dialplan se hace antes de la llamada.**

3.5.FreeSwitch en diferentes escenarios

Como comentamos previamente, FreeSwitch por su naturaleza de soft-switch o PBX, puede hacer multitud de tareas. Desde servicios de telefonía a usuarios que se encuentran en una misma LAN (mediante un servidor FS y su tarjeta de red correspondiente), hasta llamadas por internet sin coste alguno entre oficinas que comparten la misma red, o llamadas desde nuestra casa u oficina a cualquier teléfono conectado a la RTB. Puesto que las dos primeras aplicaciones comentadas tienen una configuración simple y similar a la de Asterisk, entraremos un poco más en detalle las posibilidades que ofrece FreeSwitch para conectar nuestra oficina con la RTB.

3.5.1. Conectar nuestra oficina con la RTB

Un servidor FreeSwitch puede manejar conversaciones sobre la red telefónica analógica o bien mediante la conexión a través de ADSL a un proveedor de VoIP en Internet, o mediante la conexión del servidor a una línea telefónica a través de un gateway SIP/PSTN.

En este último caso, hay dos opciones: o un dispositivo externo como el **Linksys SPA-3102**, o una **tarjeta PCI** de **Digium**, **Sangoma**, **OpenVox**, **Atcom**, etc. Sangoma también ofrece un dispositivo USB de dos FXO.

En la Figura 11 podemos ver la **pasarela VoIP Linksys SPA 3102**, que tiene un puerto FXO RJ11 para conectarlo a la roseta de teléfono, un puerto FXS para conectar un teléfono o fax analógico, un puerto Ethernet para conectarlo a la red y por último ofrece un puerto para conectarnos a Internet mediante DSL.

Figura 22: Modelo Linksys SPA 3102

En la siguiente tabla resumo las ventajas y desventajas de usar un dispositivo externo o una tarjeta PCI:

	Ventajas	Desventajas
Dispositivo Externo	-OS independiente, así FS puede funcionar en cualquier. -Más barato que las tarjetas PCI. -Proporciona un puerto FXS para conectar un teléfono analógico.	-Más cables que una tarjeta PCI (alimentación, Ethernet, PSTN). -Posible problema de eco con las conexiones SIP/PSTN.
Tarjeta PCI	-Más compacta (sólo un cable de la tarjeta a la entrada de la pared (PSTN)). -Más fiable por el nivel de la entrada.	-Requiere un ordenador de sobremesa. -Sólo funciona bajo Linux. -Posible incompatibilidad con algunas placas base.

Tabla 2: Ventajas y desventajas entre dispositivos externos y PCIs.

3.5.1.1. OpenZap

OpenZap es una biblioteca utilizada por FreeSwitch de interfaz para Sangoma, DAHDI y tarjetas PIKA. OpenZap implementa una abstracción TDM unificada tanto para la señalización (PRI, BRI, SS7, MFC-R2, etc.) como para tarjetas de E/S de APIs (API de Sangoma TDM, DAHDI, PIKA, etc.). Dependiendo de los requisitos de hardware, los controladores propios seguirán todavía siendo necesarios. Si usted tiene tarjetas Sangoma debe instalar los controladores Wanpipe primero, OpenZap entonces utilizará los controladores para comunicarse con ellas. De la misma forma se aplicará para las APIs de *Zaptel* o DAHDI.

OpenZap también ofrece servicios como la detección DTMF para las aplicaciones de usuario (en la mayoría de los casos FreeSwitch).

3.5.1.1.1. Diseño

En lo referente a *zaptel*, solo se usa como un dispositivo a través de la toma de corriente abierta en el sistema operativo para acceder a las funciones de bajo nivel como apertura, cerrado, lectura y escritura y el resto de la funcionalidad se implementa en nuestra propia capa de espacio de usuario. Para cualquier modificación, deberíamos usar la interfaz del núcleo de *Zaptel*, de forma que si por ejemplo tenemos una tarjeta que requiere modificar los drivers, podríamos trabajar siempre y cuando mantengamos la misma interfaz *zaptel*.

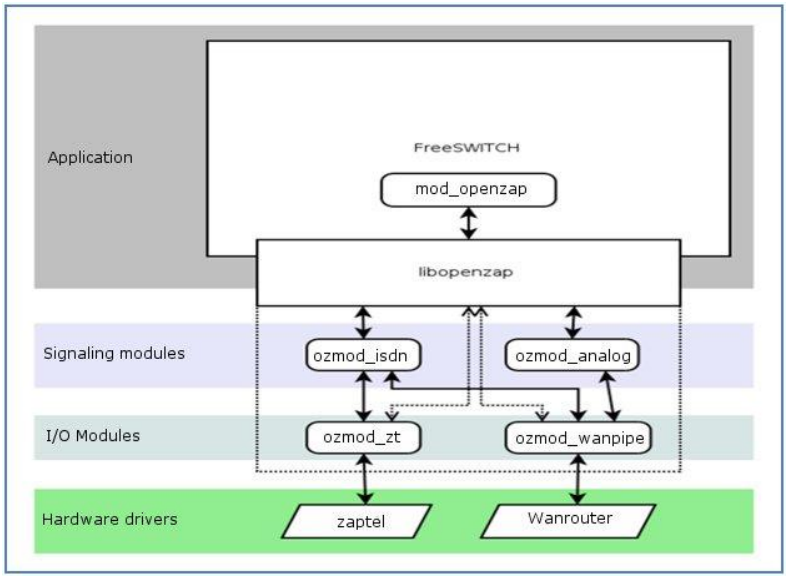

Figura 23: Diseño OpenZap.
(Fuente: http://wiki.freeswitch.org/wiki/OpenZAP)

3.5.1.1.2. Compatibilidad Hardware/Controladores

OpenZAP debe trabajar con las últimas **APIs TDM de Sangoma**. El soporte de hardware hasta el momento incluye *zaptel* y API nativa TDM de Sangoma. El código base existente incluye *mod_openzap*.

Las tarjetas **Rhino** usan *Zaptel*, sin embargo hay algunas salvedades[5].

3.5.1.1.3. Terminología FXO/FXS

Un puerto que se conecta a una línea telefónica es un puerto FXO. Si utiliza zaptel, se configura el puerto como un puerto FXO en *zaptel.cfg*. El puerto utiliza la señalización FXS, por lo que OpenZAP se configura en *openzap.conf* para usar la señalización FXS por el mismo puerto.

Un puerto que se conecta a una extensión telefónica es un puerto FXS. Si utiliza zaptel, se configura el puerto como un puerto FXS en *zaptel.cfg*. El puerto utiliza la señalización FXO, por lo que OpenZAP se configura en *openzap.conf* para usar la señalización FXS por el mismo puerto.

En resumen:

	Tipo/zaptel.cfg	Señalización/*openzap.conf*
Línea de teléfono	FXO	FXS
Extensión de teléfono	FXS	FXO

Tabla 3: Terminología FXO/FXS.

4. Mercado actual de telefonía de código abierto

En el mercado actual, Asterisk no es la única solución para la telefonía de código abierto. Otros proyectos, cada uno con sus comunidades de desarrolladores, ofrecen las funcionalidades de un PBX y aunque no son tan populares como Asterisk son soluciones perfectamente válidas. Para hacer un recorrido por las distintas posibilidades que nos ofrece el mercado, vamos a hacer una división de éste en tres niveles: plataformas de telefonía, soluciones configurables y soluciones llave en mano.

[5] Consultar apartado de interoperabilidad en la wiki de FreeSwitch http://wiki.freeswitch.org/wiki/Interop_List.

4.1.Plataformas de telefonía

En este primer nivel tenemos los proyectos que proveen la tecnología necesaria para desarrollar soluciones de telefonía. Son por tanto plataformas para desarrollar servicios de telefonía utilizados por usuarios con un alto nivel técnico como por ejemplo empresas constructores de PBX, de pasarelas VoIP de soft-phones, etc. Destacamos plataformas como:

- **sipXecs**[6]: el cual es compatible con Linux, Windows y MAC pero no soporta tarjetas para conectarse a la RTB, con lo que es obligatorio el uso de pasarelas adicionales si queremos conectarnos a ella.
- **Yate**[7](Yet Another Telephony Engine): es una solución que destaca por su flexibilidad ya que permite implementar un gran número de dispositivos tales como: clientes o servidores de VoIP, PBX, H323 Gatekeeper, etc. Escrita en C++, se puede instalar en Linux o Windows y ofrece la posibilidad de extender sus funcionalidades usando para ello lenguajes de programación como PHP, Python y Perl.
- **Callweaver**[8]: también conocido como OpenPBX, es una plataforma para desarrollar PBX derivada de Asterisk. Puede funcionar en un gran número de plataformas tales como: Linux, MacOS X/Darwin, Open/Solaris, FreeBSD, NetBSD and OpenBSD.

4.2.Soluciones gratuitas configurables

En este segundo nivel, tenemos soluciones que proveen servicios personalizados aunque permiten la adición de nuevos módulos para tener nuevas funcionalidades. Son más fáciles de configurar e instalar que las soluciones del nivel uno y vienen normalmente con una interfaz gráfica. Son todas gratuitas y entre ellas destacamos las siguientes:

Trixbox[9]: solución que implementa un IP PBX basado en Asterisk pensado para pequeñas y medianas empresa. Se distribuye en un CD con el cual instalamos en el sistema: Asterisk, Linux, FreePBX GUI, Apache, MySQL, y Perl/PHP/Phyton.

FreePBX[10]: es otro IP PBX basado en Asterisk y da nombre a la interfaz gráfica más conocida para configurarlo. Está escrita en PHP y se sirve de una base de datos MySQL.

---

[6] Para obtener más información podemos consultar su página web: http://www.sipfoundry.org.

[7] Para obtener más información podemos consultar su página web: http://yate.null.ro/pmwiki/index.php.

[8] Para obtener más información podemos consultar su página web: http://www.callweaver.org/blog.

[9] Para obtener más información podemos consultar su página web: http://www.trixbox.org.

AsteriskNow: se trata de una distribución de Asterisk que incluye además un sistema operativo Linux adaptado para funcionar con él y la interfaz AsteriskGUI para configurarlo. Esta distribución es compatible con FreePBX.

4.2.1. Soluciones de pago llave en mano

Por último, en un tercer nivel se encontrarían las aplicaciones llave en mano, las cuales proporcionan una solución completa a usuarios finales que simplemente las usan y que a cambio del desembolso económico obtienen una serie de ventajas como el soporte y mantenimiento. Aquí nos encontramos con la versión de pago de las soluciones descritas en el nivel "2". Entre ellas cabe destacar:

TrixboxPro: es la versión de pago de Trixbox y nos ofrece aplicaciones ya configuradas que nos permite tener generación de información de errores, actualizaciones automáticas, sistemas de copias de seguridad periódicas, etc.

Además su compra incluye también un servicio de soporte y mantenimiento.

Asterisk Business: como la anterior, es la versión de pago de AsteriskNow. Ofrece diferentes fórmulas dependiendo de la cual tendremos más o menos funcionalidades, tiempo de monitorización del sistema e incluso tiempo de formación para los usuarios.

5. Asterisk Vs FreeSwitch

En apartados anteriores hemos conocido las características de Asterisk y FreeSwitch. Esta sección pretende enfrentar ambas plataformas y destacar los aspectos principales y más comúnmente comentados que hacen que desarrolladores y usuarios se decanten por una de ellas.

A continuación detallaré un resumen de las experiencias que tanto desarrolladores, empresas y usuarios domésticos, hemos podido encontrar en estos sistemas, y así poder ayudar en la medida de lo posible a personas que estén pensado instalar alguna de estas herramientas y no sepan por cuál decantarse, o a usuarios actuales, para que conozcan las limitaciones o avances que son capaces de ofrecer.

[10] Para obtener más información podemos consultar su página web: http://www.freepbx.org.

5.1.Instalación y configuración

5.1.1. Asterisk

El proyecto Asterisk es el pionero en este tipo de tecnologías, o al menos, el referente que ha servido de base para muchos de sus competidores. Esta veteranía hace que tenga un mayor soporte en cuanto a información y comunidades.

Para la instalación de Asterisk existen versiones, habiéndome decantado por la versión **Asterisk 1.6.0.6**, una de las más recientes y estables. Actualmente la mayoría de la comunidad tiene instalada la versión 1.4, que según dicen es la más fiable y estable. Sin embargo ante la limitación de las pruebas que puedo realizar desde mi ordenador me he decantado por una más avanzada y suficiente para mi trabajo.

Como antes comentaba Asterisk goza de una comunidad enorme, con lo que existen gran cantidad de manuales y guías que permiten instalar Asterisk paso a paso, sin necesidad de complicaciones, a priori.

La instalación que he llevado a cabo se resume en 9 rápidos pasos[11]:

- Satisfacer los requisitos de Asterisk.
- Descarga y extracción de Asterisk.
- Pre-construcción de la configuración.
- Construcción.
- Instalación.
- Instalación de los archivos de configuración de ejemplo.
- Scripts de inicio para conseguir que Asterisk inicie al arrancar el sistema.
- Ejecución de Asterisk.
- Reinicio y verificación.

Ahora tendríamos Asterisk instalado en nuestro ordenador. Para poder probar su funcionamiento necesitamos descargar un **soft-phone**. En esta ocasión me he decantado por el softphone **X-lite**, que puede ser descargado gratuitamente

[11] El código necesario para el cumplimiento de estos pasos se puede seguir en el blog *mikeOverIP* (http://mikeoverip.wordpress.com/2009/03/11/asterisk-16-compilation-and-installation-on-debian-5-lenny/).

de la página oficial[12] y ofrece un buen funcionamiento, como he comentado anteriormente en la sección de soft-phones.

La **instalación de X-lite y configuración de X-lite** es bastante sencilla. Simplemente tenemos que descargar el paquete de la página, descomprimirlo en nuestro ordenador y ejecutarlo. Una vez abierto introducimos un usuario de prueba y la IP de nuestro servidor Asterisk y podemos realizar alguna llamada de prueba a algunos números que vienen por defecto en el plan de numeración como pueden ser el menú principal, prueba de eco, etc.

Ya tenemos instalado Asterisk debemos configurarlo y crear nuestros propios usuarios y asociarle sus extensiones y buzones de voz. Esto lo hacemos editando los ficheros ***sip.conf***, ***extensions.conf*** y ***voicemail.conf***, respectivamente[13]. Una vez hechas estas modificaciones configuramos el softphone para estos usuarios y estaríamos listos para dar nuestros primeros pasos en Asterisk.

Las limitaciones o problemas que he encontrado en la instalación y configuración de Asterisk han sido los siguientes:

- He tenido algunos fallos en la instalación por la **ausencia de librerías** que demandaba Asterisk, pero son fácilmente subsanables buscando otras alternativas en la web.
- También tuve algún problema con la **configuración del audio**, también resuelto buscando en la web.
- Aunque no se pueda considerar como un fallo, si que considero la instalación algo **tediosa**. Existen **multitud de manuales**, pero muchos de ellos **diferentes entre sí**. Es difícil una instalación y configuración total y sin problemas consultado un solo manual, sin necesidad de consultar la web. Por ellos recomiendo alternativas de instalación como **AsteriskNow**, **Elastix** o **Trixbox**, que son distribuciones basadas en Asterisk con complementos que facilitan su instalación y uso y además pueden ser cargadas e instaladas mediante CD, facilitando en gran medida el proceso.
- Otra dificultad en Asterisk es su manejo. Al usar el intérprete de comandos hace la herramienta mucho menos intuitiva y poco entendible. Aconsejo el uso de interfaces gráficas como **FreePBX**, que puede disminuir sensiblemente la dificultad de configuración y forma de usarlo.

[12] Página web oficial: http://www.counterpath.com/x-lite.html

[13] Los pasos la configuración de estos ficheros se puede consultar en la página http://www.voipforo.com/asterisk/asterisk_primeros_pasos.php.

5.1.2. Instalación y configuración FreeSwitch

FreeSwitch se presenta como la gran alternativa a Asterisk. Es un proyecto **muy joven**, con lo que no tiene tanto soporte como su competidor, pero eso sí, tiene una comunidad muy activa, que por tanto hace cada vez más fuerte la herramienta y facilita su uso al resto de las personas.

Al igual que con Asterisk, existen varias versiones de esta plataforma, habiéndome decantado por las más reciente que se puede descargar de su página oficial, la versión **FreeSwitch 1.0.6.**

Puesto que no tiene un soporte tan grande como Asterisk, ni en cuanto a información o fama, FreeSwitch compensa este desnivel con una página oficial con gran cantidad de información y una **wiki excepcional**. Esta wiki que comento ha sido la que he seguido para realizar la instalación y configuración de la herramienta, **sin necesidad de buscar por la web otras alternativas como sucedía con Asterisk**.

En la misma wiki tenemos varias alternativas de instalación, algunas más lentas y seguras, otras usando repositorios, y otra rápido y simple. Me he decantado por la que usa **Git** como repositorio.

En cuanto a los pasos de instalación son similares en ambas plataformas. Para FreeSwitch omito estos pasos para no repetir información[14].

Una vez lo tenemos instalado podemos realizar de forma sencilla una llamada de prueba para comprobar que funciona correctamente. Para ellos es necesaria la descarga e instalación de un softphone. Al igual que con Asterisk he elegido **X-lite**. No me detendré en temas relativos a su instalación y configuración ya que lo comenté anteriormente con Asterisk. Una vez lo configuramos podemos realizar esta llamada por defecto que he comentado y así ver que todo va correctamente.

Para su configuración y puesta en funcionamiento tenemos en la misma wiki de FreeSwitch una guía (***Getting start guide***) en la que se nos muestra los archivos a modificar y algunas configuraciones por defecto que pueden facilitarnos la instalación. Recomiendo la lectura de esta guía, sencilla y clara, a cerca de la iniciación con FreeSwitch.

[14] Toda la secuencia de código e instrucciones para su ejecución podemos verlo en la wiki de FreeSwitch (http://wiki.freeswitch.org/wiki/Installation_Guide).

Las limitaciones o problemas que he tenido en la instalación y configuración de FreeSwitch han sido las siguientes:

- Algún problema con las librerías, solucionado buscando en los foros de la comunidad y realizando una actualización.
- Igual que en Asterisk, **recomiendo usar una interfaz gráfica**, ya que el funcionamiento por medio de la línea de comandos es poco vistoso y dificulta el uso de la herramienta. Recomiendo **FusionPBX** (recientemente realizada para FreeSwitch), **FreePBX** (bastante completa y adaptada hace poco para FS), o **WikiPBX** (sencilla y fácil de manejar). Todas realizadas en diferentes lenguajes de programación.

5.1.3. Conclusión

Después de haber instalado y configurado ambos sistemas, he de decir que **las instalaciones no suponen un grado importante de dificultad** estando ambas al mismo nivel. En este tipo de **configuraciones sencillas** y con **pruebas simples ambas plataformas proporcionan un buen funcionamiento** y **no destaca especialmente una respecto a otra**. Será en entornos empresariales, con grandes exigencias, y con importantes requisitos o modificaciones de código cuando podremos apreciar las diferencias en que hacen una de estas centralitas la mejor candidata sobre la otra.

En el punto siguiente nos centraremos en este tipo de requisitos y analizaremos la que según los profesionales del sector puede ofrecer un mejor rendimiento según los requerimientos que necesitemos.

5.2.Análisis

Para introducir la comparativa de ambas plataformas comentaré brevemente cuáles fueron los motivos que impulsaron la creación de FreeSwitch ante un existente y afamado Asterisk. A continuación analizaremos tanto Asterisk como FreeSwitch, resumiendo las principales características que podrían hacer que nos decantásemos por uno u otro.

5.2.1. ¿Por qué crear FreeSwitch?

El creador de FreeSwitch, Anthony Minessale comenzó como desarrollador de Asterisk en 2003 llegando a tener gran conocimiento de éste. Comentaré su experiencia ya que nos ayudará a entender mejor los pilares de estas plataformas, así como a comprender la visión que tiene un desarrollador.

5.2.1.1. Problemática con Asterisk

La primera vez que Anthony Minessale tuvo su primer **"callejón sin salida"** con Asterisk fue desarrollaba una infraestructura para usar sus colas de llamadas entrantes. Se podía llamar desde un número PSTN sobre un T1 y unirse a una cola de llamadas, donde los agentes de Asterisk, que también llamaban podían atender las llamadas. Todo parecía estupendo hasta que se produjo este primer fallo y sembró la duda en el desarrollador.

Después de esta mala experiencia vinieron otras muchas: bloqueos en el administrador, en las colas de la aplicación, *Segmentation Fault*,…

Incluso tuvo que **abandonar algunas características** en sus desarrollos porque simplemente no funcionarían bien según el diseño de Asterisk.

Asterisk utiliza un diseño modular, donde un núcleo central carga objetos compartidos para extender la funcionalidad con trozos de código llamados "módulos". El núcleo de Asterisk es un **modelo de hilos,** pero muy conservador, de forma que **sólo los canales de ejecución y creación tienen hilos[15].** Este esquema de hilado fue uno de los **factores de motivación para su reescritura.**

Asterisk también usa **listas vinculadas** para gestionar sus canales abiertos. Son una práctica de programación útil, pero en aplicaciones de hilos pueden llegar a ser muy difíciles de manejar. Se necesita el uso de ***mutexes***, una especie de semáforo para hilos para asegurarse de que sólo cada vez un hilo tiene acceso de escritura a la lista, o se arriesga a un hilo a arrancar un enlace de la lista, mientras que otro la recorre[16].

El núcleo de Asterisk **vincula las dependencias de algunos de sus módulos**, lo que significa que la aplicación no se iniciará si uno o varios módulos específicos no están presentes porque el núcleo esté utilizando directamente una parte del código binario del objeto compartido del módulo.

[15] Este modelo de hilos implica que la fase B de cualquier llamada sólo puede operar en el mismo hilo de la fase A, y cuando algo sucede, como una transferencia de llamadas, el canal debe ser transferido a un modo de hilos (*threated mode*), que muchas veces incluye una práctica llamada *enmascaración de canal*, un proceso en el que todos los elementos internos de un canal son arrancados de un objeto de memoria dinámica y trasladados a otro. Debido a esto, a menudo se verán 3 o 4 canales para una sola llamada durante una transferencia de llamada.

[16] Esto también conduce a situaciones horribles en que un hilo puede estar destruyendo o enmascarando un canal mientras que otro accede a él, lo que provocará en un fallo de segmentación y por supuesto, significa en la mayoría de los casos todas las llamadas se perderán.

Por problemas como estos, Anthony Minessale llegó a la conclusión de que la solución era reescribir el código creando una nueva versión de Asterisk. Puesto que no tuvo apoyo, decidió crear FreeSwitch

5.2.1.2. Creación de FreeSwitch

El enfoque principal en FreeSwitch era **empezar desde el núcleo** y hacer caer toda la funcionalidad común en virtud de la campana de forma que forma una pirámide hacia los niveles superiores de la aplicación.

Se optó por usar un **diseño modular teniendo cada canal su propio hilo** sin importar lo que esté haciendo. El hilo usará una función de máquina de estado para recorrer su camino a través del núcleo.

Puesto que todos los canales funcionan con su propio hilo, si hay ocasiones en las que se necesita interactuar con ellos, se usa bloqueo de lectura/escritura, de manera que **los canales se pueden localizar con un algoritmo de hash en lugar de una lista enlazada**, donde en absoluto garantiza que el canal no pueda estar accesible o desaparecer mientras que un hilo exterior tenga referencia a él mismo.

La mayoría de las funciones y los objetos proporcionados por el núcleo FreeSwitch están protegidos de la persona que llama, obligándoles a utilizar la forma en que fueron diseñados. **Cualquier concepto que se puede ampliar o sea proporcionado por un módulo tiene una interfaz específica que es usada de manera que esa funcionalidad de interfaz, y por tanto el núcleo, no tenga ninguna vinculación de dependencia en cualquiera de sus módulos.**

Hay una clara capa de API con las funciones del núcleo que están en el fondo y una cantidad de funciones en cada capa superior que disminuye a medida que aumenta la funcionalidad.

Todos los módulos FreeSwitch trabajan juntos y se comunican entre sí sólo a través del núcleo de la API y el sistema de eventos internos (evitando así cualquier comportamiento no deseado de módulos)**.**

El **sistema de eventos** en FreeSwitch fue diseñado para realizar un seguimiento en la medida de lo posible. Se diseñó bajo el supuesto de que la mayoría de los usuarios del software se conectarían a FreeSwitch de forma remota o el usando un módulo personalizado para recopilar datos de

la llamada. Por lo tanto, **todo lo importante que sucede en FreeSwitch resulta del disparo de un evento.**

Otro concepto importante con el que se creó FreeSwitch fue el **registro centralizado XML**. Utilizando **módulos de enlace XML** se puede enlazar el módulo de búsquedas en el registro XML y, en tiempo real, reunir la información solicitada y devolverlo a la persona que llama en lugar de los datos estáticos en el archivo. Esto hace que sea posible hacer registros SIP puramente dinámicos, buzones de voz dinámicos y configuraciones dinámicas de un clúster usando el mismo modelo que un navegador web y una aplicación CGI.

Éstas son sólo algunas de las características con las que se creó FreeSwitch, pero que responden perfectamente a la pregunta que da título a este apartado.

5.2.2. ¿No es FreeSwitch un soft-switch y Asterisk una PBX?

¿No es FreeSwitch un soft-switch y Asterisk PBX? ¿Cuál es la diferencia? [10] Muchos nos hemos hecho esta pregunta y quizá hayamos pensado que si realmente FreeSwitch no es una PBX, probablemente al trabajar como tal no será tan efectiva como Asterisk o no cumplirá nuestros requisitos…Si hemos pensado esto estamos equivocados…A continuación explico qué es exactamente un soft-phone, qué es una PBX y por qué FreeSwitch puede desempeñar sin problemas una gran labor como PBX.

"Un *PBX* es una entidad que permite a una empresa privada tener su propia mini compañía de teléfono prestando servicios como buzón de voz, extensiones y conferencias a los teléfonos. El objetivo principal de una PBX es para varios teléfonos, encontrarse y comunicarse.

Un soft-switch es una aplicación de software que puede conectar las líneas telefónicas de una red a otra, a menudo encaminando de las llamadas de un protocolo a otro o a un punto de terminación, como una PBX.

FreeSwitch tiene el potencial de funcionar también como una PBX pero no tiene el mandato de hacerlo. Piense en ello como una aplicación de nivel más bajo que un PBX. Es posible cargar varios módulos en FreeSwitch para hacer que se comporte exactamente como un conjunto de aplicaciones de PBX. **Esto es completamente más fácil que tratar de hacer una única PBX monolítica que se comporte como un switch, especialmente cuando gran parte de las**

funcionalidades de PBX están permanentes en el núcleo de la aplicación PBX."[17]

5.2.3. ¿Por qué elegir FreeSwitch?

A continuación resumo algunas de las principales ventajas que tiene FreeSwitch frente a Asterisk:

- **Código fuente limpio** → Hecho desde cero. Al haber comprobado las limitaciones de Asterisk, FreeSwitch se construyó de forma mucho más escalable. Asterisk es más antiguo y mucho código es una colección de añadidos que lo hacen mucho más complejo[18]. Esto implica **mejor estabilidad y escalabilidad** para FreeSwitch, y **mejor control** (es más sencillo formar *devs* y reparar *bugs*) para las empresas y usuarios.

- **Licencias de código** → Al tener licencia MPL, **se pueden usar librerías *Open Source* de calidad y no tener que reinventar o reescribir código** cada vez que se quiere implementar una función. Por ejemplo, no implementaron un *chan_sip* desde cero como en Asterisk, sino que usaron una librería (*sofia_sip*) hecha por Nokia y cumpliendo totalmente con las distintas RFCs involucradas. Asterisk con el sistema de Dual-Licensing Digium-GPL, lo tiene mucho más limitado y se piensa que FreeSwitch avanzará más rápido aún teniendo 10x menos recursos de desarrollo.

- **Funcionalidades** → Siguiendo con el ejemplo SIP, que es uno de los componentes esenciales para las empresas, y por las razones anteriores, FreeSwitch ya tiene **muchas más funciones disponibles que no tiene Asterisk** (chat, registro múltiple, múltiples dominios, múltiples UAs,...) [11].
  También **soporta más codecs** (CELT a 48KHz), **más funciones de voz** (VAD, CNG,...), **mejor integración con múltiples idiomas para scripting/IVRs** (C, LUA, PHP,...), **mejor integración con TTS/ASR** (protocolo MRCP,...), etc.

- **Rendimiento** → **Multiplicado al menos por 5 (sino 10 o más) en entornos similares**. Anthony Minessale logró tener 10001 llamadas

[17] Extraído del FAQ de FreeSwitch
[18] Ver *chan_sip.c* en la web.

simultáneas con un simple Quad Core[19], cuando a menudo se pelea para obtener 300 con Asterisk (en benchmarking similar con sólo música en espera) [12].

- **Mucho más que una PBX** → FreeSwitch es mucho más que una PBX. Podríamos decir que es una librería, lo que significa que nos permite desarrollar desde un softphone hasta funcionalidades completas de PBX, proyectos como FSComm, etc., abriendo un importante abanico de utilidades.

5.2.4. ¿Por qué elegir Asterisk?

Acabamos de leer los puntos fuertes que hacen destacar a FreeSwitch sobre Asterisk, pero Asterisk por su madurez y gran soporte también tiene importantes características que FreeSwitch aún no ha conseguido superar y las enumero a continuación:

- **Tarjetas TDM** → De momento **Asterisk + *chan_dahdi*** sigue siendo superior a **FreeSwitch + FreeTDM/OpenZap**, para tarjetas TDM (analógicas sobre todo). Sin embargo, Sangoma piensa hacer frente a su competidora y se puso en marcha con el desarrollo de FreeTDM, el cual cada día está cada vez más avanzado. Se piensa poder usarlo en un futuro a corto/medio plazo en las empresas.

- **Potencia de mercado** → Con la madurez de Asterisk y su fuerte potencial adquirido con los años, tiene mucha publicidad y buen soporte con las versiones comerciales de Asterisk (aunque sufren de los mismos problemas). El camino de Asterisk en solitario ha hecho que para muchos sea la única referencia de PBX open source. Sin embargo, **en entornos de operadores no tiene tan buena publicidad** porque suele ser relativamente difícil hacer que funciona de manera correcta. Asterisk siempre contará con el respaldo de **Digium**, lo cual es una importante ventaja (aunque FreeSwitch ahora tiene apoyo de Sangoma y Barracuda Networks entre otros…).

- **Funcionalidades** → La comunidad Open Source de Asterisk es mucho mayor que la de FreeSwitch lo que hace que muchas personas piensen primero en crear sus aplicaciones con Asterisk. Probablemente

[19] Más información en el enlace http://www.viddler.com/explore/cluecon/videos/2/.

algunos desarrollos estén disponibles sólo con Asterisk, aunque por ahora es complicado encontrar alguna aplicación que no exista o no tenga una solución sencilla con FreeSwitch.