3. Asterisk

3.1. Introducción

Asterisk es una aplicación de centralita telefónica PBX bajo licencia GPL. Fue desarrollada por Mark Spencer, entonces estudiante de ingeniería informática en la Universidad de Auburn, Alabama. Mark había creado en 1999 la empresa 'Linux Support Services' con el objetivo de dar soporte a usuarios de Linux. Para ello necesitaba una centralita telefónica, pero ante la imposibilidad de adquirirla dados sus elevados precios, decidió construir una con un PC bajo Linux, utilizando lenguaje C. Posteriormente 'Linux Support Services' se convertiría en el año 2002 en 'Digium', redirigiendo sus objetivos al desarrollo y soporte de Asterisk.
Digium es la empresa que promueve e invierte en el desarrollo de código fuente y en hardware de telefonía de bajo coste que funciona con Asterisk. El sistema operativo que 'monta' puede ser cualquiera de las distintas distribuciones de Linux, con o sin hardware que conecte a la red pública de telefonía (PSTN: Public Service Telephony Network). Permite conectividad en tiempo real entre las redes PSTN y redes VoIP.

Asterisk es mucho más que una centralita telefónica. Se pueden hacer más cosas con facilidad como:

- Conectar empleados trabajando desde cualquier ubicación sin más que habilitar conexiones de banda ancha.
- Conectar oficinas con la PBX central sin importar su ubicación, mediante un enlace por Internet o por una red IP privada.
- Proveer a los usuarios de buzón de voz, integrándolo con una web y/o sus e-mails.
- Construir menús de respuesta automática por voz, que pueden ser conectados a un sistema de aplicaciones internas.
- Dar acceso al sistema para usuarios que viajan, conectándose a cualquier acceso a Internet (un aeropuerto, un hotel o puntos libres de conexión).

Asterisk es capaz de prestar servicios que antes solo eran alcanzables mediante pago con algún proveedor de telefonía, como:

- Música en espera con archivos mp3.
- Colas de llamada donde agentes de forma conjunta atienden las llamadas.
- Sintetización de la conversación (text-to-speech).
- Registro detallado de llamadas (call-detail-records) para sistemas de tarificación.
- Integración con reconocimiento de voz.
- Interfaces con líneas telefónicas normales, ISDN en acceso básico (2B+D) y primario (30B+D).

Digium es la creadora y desarrolladora de Asterisk, aunque existe la libertad de modificar su contenido (filosofía del software libre) y depurar y reportar errores, lo que provoca que muchos programadores hayan contribuido con nuevas características y mejoras. Digium es fundada en Huntsville, Alabama, son los creadores de las tarjetas de telefonía PCI, que ofrecen una excelente relación coste/beneficio para el transporte de voz y datos a través de las redes analógicas, RDSI o Ethernet. Para el control de estas tarjetas se comenzó el denominado 'proyecto ZAPATA' que se encargaba del desarrollo del hardware de Digium. Es interesante resaltar que el hardware también es abierto y puede ser producido por cualquier empresa.

- ***Proyecto ZAPATA***

El proyecto ZAPATA fue conducido por Jim Dixon, responsable del desarrollo hardware de Digium. La implicación de empresas ajenas a Digium se hace notar cada vez más, por ejemplo, la placa con 4 E1/T1 es producida por Digium, Sangoma y también por Varion.
La historia del proyecto zapata se resume así:

Hace 20 ó 25 años, la AT&T (American Telephone & Telegraph) comenzó a ofrecer una aplicación informática que permitía a los usuarios personalizar la funcionalidad de su sistema de buzón de voz y recepción de llamadas, el cual recibió el nombre de Audix. Audix corría sobre Unix y a un coste muy alto, como todo en telefonía hasta el momento, con una funcionalidad bastante limitada.
En un intento de cambio, algunos fabricantes vieron que con una tarjeta colocada en un PC con MSDOS y conectada a una línea telefónica (FXO solamente) conseguían comunicaciones. Las tarjetas no tenían una calidad tan buena comparada con las actuales, y muchas terminaron como 'secretarias electrónicas' aunque de poca calidad.
Se produjeron nuevas tarjetas de telefonía, aunque a precios altos, y las compañías continuaron invirtiendo mucho dinero en investigación y desarrollo de las mismas. Al final, con la fuerte inversión de muchos fabricantes, las tarjetas de telefonía consiguieron tener mucha capacidad de procesamiento en forma de DSP (procesador de señales digitales). El poder de procesamiento de los microcomputadores continuó creciendo y Jim Dixon decidió seguir mejorando la primera línea de investigación de las tarjetas (ahora con procesadores más potentes) para tratar de reducir costes en su fabricación. De esta forma compró una tarjeta Mitel89000C 'ISDN Express Development Card' y escribió un driver para el sistema operativo FreeBSD. La tarjeta ocupó poco procesamiento de un Pentium III 600Mhz, y comprobó que si no fuese por la limitación de E/S (la tarjeta trataba de forma ineficiente la E/S) podría atender de 50 a 75 canales. Como resultado de este acontecimiento, Jim Dixon compró lo necesario para crear un nuevo diseño de tarjeta que usase de forma más eficiente la E/S. Consiguió que un PC gestionara sin problemas dos T1 (48 canales) de datos transferidos sobre el bus entre memoria y el microprocesador. Con esto, ya tenía las tarjetas preparadas para la venta (se vendieron unas 50). Colocó el diseño completo, incluyendo archivos de los planos de la tarjeta, en la web.

El nombre viene a raíz de la sensación de concepto revolucionario que le causó a Jim Dixon y decidió colocar un nombre inspirado en el revolucionario mexicano Emiliano Zapata, decidiendo llamar a la tarjeta 'ZAPATA'. Escribió un driver completo y lo colocó en la red. La respuesta de la mayoría de los internautas que mostraron interés fue: 'Muy bien, muy bien pero... tiene para Linux?' Jim Dixon comenta esta situación de manera personal como:

'Personalmente yo nunca había visto Linux antes, pero fui rápidamente al Fry's (una tienda enorme de productos electrónicos, famosa en E.E.U.U.) y compré una copia de Linux Red Hat 6.0. Eché un vistazo a los drivers y usé el Vídeo Spigot como base para traducir el driver de FreeBSD a Linux. Mi experiencia con Linux no fue buena y comencé a tener problemas en el desarrollo de los módulos cargables del kernel. De cualquier forma lo colgué en la red sabiendo que algún gurú en Linux se reiría de él y tal vez me ayudara a reformarlo en un 'Linux apropiado'. En 48 horas recibí un e-mail de un sujeto de Alabama (Mark Spencer), que se ofreció para hacer exactamente esto. Me enteré que él decía que tenía algo que sería perfecto para a todo esto en conjunto (Asterisk).'

JIM DIXON

En ese momento Asterisk era un concepto funcional, porque no tenía una manera real de funcionar de forma práctica y útil. La unión del sistema de telefonía Zapata y el diseño de bibliotecas de hardware/driver e interfaces, permitirían a Asterisk crecer para convertirse en una centralita telefónica (PBX) real que pudiera hablar con teléfonos y líneas reales.
Así, Mark era brillante en VOIP, redes, en la parte interna del sistema etc., y tenía un gran interés en teléfonos y telefonía, pero tenía experiencia limitada en sistemas de telefonía y en cómo estos

funcionaban, particularmente en el área de interfaces de hardware. Ésa era la faceta en la que intervenía Jim Dixon. Ambos estuvieron proporcionándose información e implementando código de drivers y de PBX's. Se hizo un buen equipo trabajando en un objetivo común. Tras la tarjeta ISA, Jim Dixon diseñó la 'Tormenta 2 PCI Quad T1/E1', que Mark vende como Digium T400P y E400P, y ahora Varion está vendiendo como V400P (Ambos T1 e E1).

Todos los archivos del proyecto (incluyendo foto y archivos de planos) están disponibles en zaptelephony.org para uso público:

http://www.zapatatelephony.org

Gracias al trabajo dedicado de Mark y de otras personas, los drivers de Zaptel y el software de Asterisk, son tecnologías que vienen de un largo tiempo atrás y que están creciendo y mejorándose día tras día.

Asterisk trae un cambio profundo en todo el mercado de las telecomunicaciones y voz sobre IP. La mayor razón para elegir Asterisk empieza por ser un software potente y, sobre todo, económico. Muchas de las causas que están haciendo grande a Asterisk son:

- ***Reducción extrema de costes***

Comparando una PBX tradicional con Asterisk la diferencia puede ser pequeña, principalmente por el coste inicial de hardware y los teléfonos IP. Aunque comparar una central analógica de cuatro líneas FXO y 16 ramales con Asterisk como PBX digital no es del todo justo. Una PBX analógica con una línea, unas 10 extensiones y aplicaciones de interfaz gráfica para recopilación de estadísticas puede alcanzar costes que superan en 10 a los de una PBX con Asterisk.

- ***Tener control del sistema telefónico***

Este es uno de los beneficios más característicos. No es necesario que alguien configure nuestra PBX, la puede configurar uno mismo. Existe total libertad a la hora de programar cualquier operación hasta el nivel requerido, a fin de cuentas es LINUX y es libre.

- ***Ambiente de desarrollo fácil y rápido***

Asterisk puede ser programado en C con las API's (Application Programming Interface - Interfaz de Programación de Aplicaciones) nativas, o en cualquier otro lenguaje usando AGI (Asterisk Gateway Interface).

- ***Rico y abundante en recursos***

Existen pocos recursos para equipamientos PBX (ya sean analógicos o digitales) vendidos en el mercado que no puedan ser creados en Asterisk. Ya se puede encontrar todo lo que tiene un PBX tradicional, e incluso añadir más recursos difíciles (o imposibles) de implementar hasta ahora.

- ***Es posible proveer contenido dinámico por teléfono y es código abierto***

Como Asterisk es programado con C u otros lenguajes de dominio de la mayoría de los programadores, las posibilidades de proveer contenido dinámico por teléfono no tienen límites. La adopción de nuevos usuarios es muy rápida, millares de preguntas cuestiones y relatos de problemas son enviados todos los días en numerosos foros de discusión.

- ***Limitaciones de la arquitectura de Asterisk***

Asterisk usa una CPU de servidor para procesar los canales de voz, en vez de tener un DSP (procesador de señales digitales) dedicado a cada canal. Esto permitió que el coste se redujese

para las placas RDSI E1/T1 pero hace que el sistema sea muy dependiente de la configuración de la CPU.

- ***Plan de numeración flexible y poderoso***

La mayoría de las centrales ni siquiera disponen de la posibilidad de utilizar la ruta de menor coste. Con Asterisk este proceso es simple y práctico. Se puede desviar una llamada por rutas distintas en función de la hora o la densidad de tráfico de forma fácil, rápida y sencilla. Las restricciones en llamadas salientes y/o entrantes no son una tarea tediosa.

ARQUITECTURA DE ASTERISK

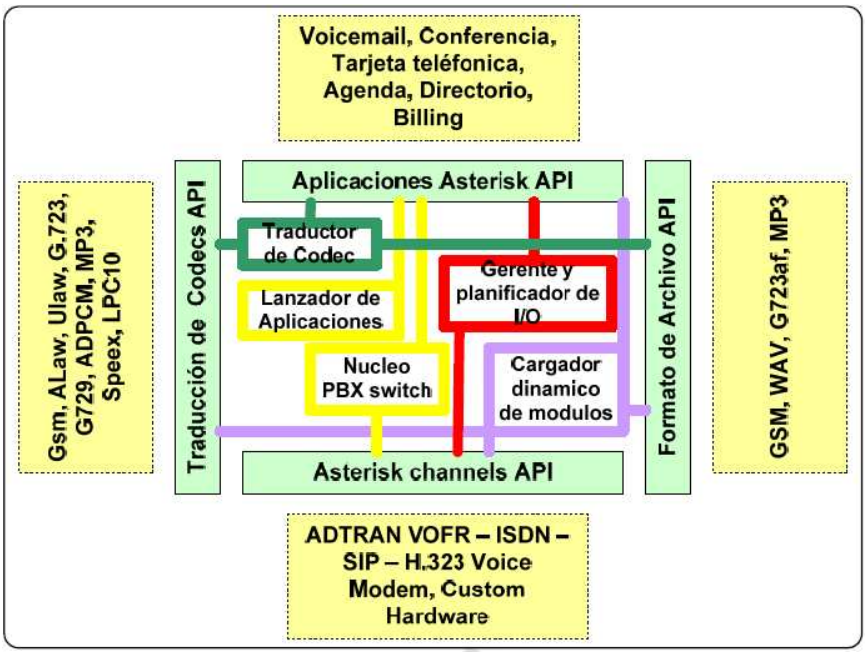

Figura 3.1.1: Arquitectura de un sistema Asterisk

La '***Figura 3.1.1***' muestra la arquitectura básica de Asterisk. Los conceptos básicos relacionados con este esquema son los canales, los códecs y las aplicaciones.

- ***Canales***

Un canal es el equivalente a una línea telefónica en la forma de un circuito de voz digital. Cada canal se compone, generalmente, de una señal analógica con alguna combinación de Códec y protocolos de señalización (GSM con SIP, G.711 ley μ con IAX). En un principio las conexiones de telefonía eran siempre analógicas y por eso, más susceptibles a ruidos y ecos. Actualmente, buena parte de la telefonía migró al sistema digital, donde la señal analógica es codificada en forma digital usando normalmente MIC (Modulación del Impulso Codificado). Esto permite que un canal de voz sea codificado en 64 Kbps sin ser comprimido.

- ***Códecs y Conversores de Códec***

Se trata de encaminar tantas llamadas como sea posible en una red de datos. Para ello, codificamos la señal de alguna manera de forma que use menos ancho de banda. Este es el papel del códec (Coder/Decoder), codificar y decodificar la señal durante el recorrido por los canales de datos.
Algunos códecs, como el g.729, permiten codificar a 8 Kbps, esto es, una compresión de 8 a 1.

Asterisk soporta los siguientes códecs:

- G.711 ulaw (ley μ: usado en EUA) – (64 Kbps).
- G.711 alaw (ley A: usado en Europa y Brasil) – (64 Kbps).
- G.723.1 – Modo Pass-through
- G.726 - 32kbps en Asterisk1.0.3, 16/24/32/40kbps
- G.729 – Precisa adquisición de licencia, a menos que este siendo usando en modo pass-through.(8Kbps)
- GSM – (12-13 Kbps)
- iLBC – (15 Kbps)
- LPC10 - (2.5 Kbps)
- Speex - (2.15-44.2 Kbps)

- ***Protocolos***

Enviar datos de un teléfono a otro es más fácil cuando los datos encuentran un camino definido para el destino. Desafortunadamente esto no sucede así, es preciso un protocolo de señalización para establecer las conexiones, determinar el punto de destino, y también cuestiones relacionadas con la señalización de telefonía, como el tono y tiempo de ring, identificador da llamada, desconexión etc. Actualmente, es común el uso de SIP (Session Initiation Protocol), aunque también existen otros protocolos como H.323, MGCP y más recientemente IAX que es excepcional cuando se trata de trunking y NAT (Network Address Translation). Asterisk soporta:

- SIP
- H323
- IAX y IAX2
- MGCP
- SCCP (Cisco Skinny).

- ***Aplicaciones***

Para interconectar las llamadas de entrada con las de salida u otros usuarios de Asterisk, son usadas diversas aplicaciones como 'Dial' (quizá las más importantes), por ejemplo. La mayor parte de las funcionalidades de Asterisk son creadas en forma de aplicaciones como son el 'VoiceMail' (buzón de voz) o 'Meetme' (conferencia) entre otras. Podemos ver las aplicaciones disponibles en Asterisk usando el comando *'show application'* en la interface de línea de comando de Asterisk (***Figura 3.1.2***).

Figura 3.1.2: Aplicaciones Asterisk

Existen aplicaciones que se pueden añadir a partir de archivos asterisk-addons y de terceros vía internet.

3.2. Configuración de Asterisk

Esta sección explica cómo instalar y configurar la centralita a partir del software Asterisk más puro. Para ello, lo primero será buscar los paquetes de instalación e instalarlo. Más adelante se verá cómo configurar básicamente la PBX con algunos ejemplos. Asterisk funciona en muchas plataformas y sistemas operativos, pero nosotros vamos a usar una única plataforma y distribución de Linux.

3.2.1. Instalación

Para la instalación del sistema PBX solo se necesita un PC con unos requisitos mínimos que deben cumplirse. Asterisk puede abusar del procesador de la máquina ya que lo usa para el procesamiento de los canales de voz. Si se montase un sistema complejo con carga elevada habría que tener en cuenta este concepto. Para construir una PBX, con un procesador compatible con Intel que sea mejor que un Pentium 300Mhz con 256 MB RAM sería suficiente. Asterisk no requiere mucho espacio en disco, sólo cerca de 100 MB compilados más código fuente, buzón de voz, grabaciones, etc. Valdrían unos 6 Gb. Hay que reseñar que las tarjetas para comunicación con redes analógicas son PCI y, por tanto, la máquina en la que se vaya a instalar Asterisk debe disponer de ranuras libres. Además, sería bueno verificar las instrucciones de la placa-base para determinar si las ranuras PCI soportan estas tarjetas. El hardware necesario para Asterisk no tiene por qué ser demasiado complejo, aunque lo que sí es importante es que disponga de una buena tarjeta de red. Se debe comprobar que no existe conflicto de interrupciones (IRQ) entre tarjetas, ya que es una fuente potencial de problemas de calidad de audio en Asterisk. Una manera de liberar IRQ's es deshabilitar desde la BIOS lo que no fuera necesario.
Asterisk es una PBX basada completamente en software. Esta funcionalidad, que durante mucho tiempo se hacía utilizando circuitos electrónicos de conmutación, ha empezado a desarrollarse en software, haciendo los equipos más flexibles, configurables y baratos. La instalación de Asterisk en GNU/Linux no difiere mucho de la instalación de cualquier otro servicio. Antes de instalarlo veremos cómo está constituida la arquitectura general de Asterisk. En la '***Figura 3.2.1***' se muestran los conceptos a priori más importantes.

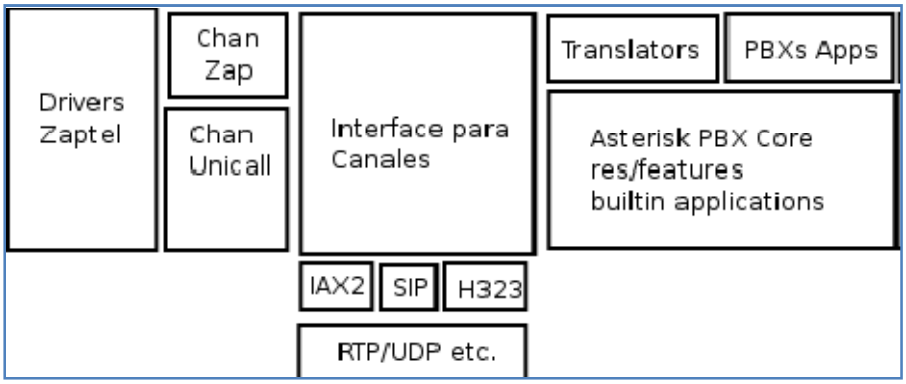

Figura 3.2.1.: Estructura del software Asterisk

Si el deseo fuera únicamente instalar Asterisk para servicios puramente VoIP, la parte del diagrama que involucra a chan_zap, chan_unicall y los drivers de zaptel desaparecerían. Sin embargo normalmente es necesario conectarse a redes tradicionales como la red telefónica pública conmutada (RTPC). Por esta razón hacen falta algunas dependencias más.

Los siguientes paquetes son necesarios para nuestra instalación:

- *sqlite 2.x (utilizaremos el sencillo manejador de CDR para SQLite)*
- *zaptel-1.2.5 (drivers para el funcionamiento de tarjetas telefónicas PCI)*

- *asterisk-1.2.7.1*
- *X-Lite de Counterpath (o cualquier otro softphone SIP e IAX)*

Para instalarlo a mano, hay que descomprimir los archivos .tar.gz con el comando:
tar -xvpzf <nombre del archivo>

Para el caso de zaptel el siguiente comando lo instalará:
make linux26
make install

Asterisk necesita:
make install

La compilación puede tomar más o menos minutos dependiendo de la velocidad del procesador. Los archivos de configuración y directorios más importantes que se instalarán son:

- ***/etc/asterisk***

En este directorio se encuentran todos los archivos necesarios para configurar la gran cantidad de servicios que Asterisk contiene. Tomaría mucho tiempo revisar todos estos servicios, por lo que se comentarán los más importantes.

1) *asterisk.conf:* configuraciones generales de la ubicación de directorios de configuraciones, módulos compilados, voicemails, etc. En general es buena idea no modificar estas configuraciones, salvo casos especiales.
2) *cdr.conf*: Configuraciones referentes al 'Call Detail Record'. Los CDR son sumamente importantes para las compañías telefónicas. Modificar datos en este archivo puede repercutir en la integridad de los CDR si no se está seguro de lo que se hace. Si la instalación es únicamente de prueba, o los CDR no son materia importante, mejor no editarlo.
3) *codecs.conf:* A menos que se quieran hacer cosas especiales con la forma en la que los códecs se comportan, es mejor no modificar este archivo.
4) *extconfig.conf:* Archivo para mapear archivos de configuración hacia tablas en alguna base de datos, de forma que no es necesario guardar las configuraciones.
5) *extensions.conf:* Tal vez el archivo más importante de Asterisk. En este archivo se toman las decisiones de encaminamiento de las llamadas.
6) *features.conf:* Este archivo es también muy importante. Permite habilitar y configurar servicios genéricos de una PBX como la transferencia asistida y monitoreo de llamadas.
7) *iax.conf*: Importante archivo para el funcionamiento del canal chan_iax que le permite a Asterisk interactuar con otros dispositivos IAX, esto es, donde se configuran las extensiones, incluyendo otros PBX Asterisk.
8) *indications.conf:* Configuraciones para los grupos de frecuencias a utilizar para la indicación del proceso de las llamadas. Los defaults suelen ser suficiente.
9) *logger.conf:* Define el nivel de verbosidad (cantidad de comentarios) que deben tener los mensajes de registro y a dónde deben ser enviados.
10) *modules.conf:* Archivo sumamente importante. Determina qué módulos serán cargados por Asterisk al iniciar.
11) *sip.conf:* Análogo del archivo iax.conf para el protocolo SIP.
12) *zapata.conf:* Configuración de los canales Zap. Las configuraciones de este archivo deben coincidir con el hardware instalado y la configuración del driver zaptel.

- ***/var/log/asterisk***

Cuando hay problemas, este es el lugar en donde hay que buscar. En esta carpeta se encuentran los archivos de registro de las operaciones de Asterisk. Estos son los archivos más importantes que se pueden encontrar:

1) *cdr.db:* Este archivo se encuentra disponible si se cuenta con el CDR handler para la base de datos SQLite. El archivo contiene la base de datos de los registros de las llamadas.
2) *event_log:* Registro de eventos sucedidos en el PBX.
3) *full:* Creado con la intención de contener todos los mensajes de debug del sistema.
4) *messages:* contiene un listado de los mensajes de warning, debug y demás niveles de registro.
5) *queue_log:* Archivo utilizado principalmente por la aplicación app_queue.

- ***/var/lib/asterisk***

Directorio con archivos de audio, llaves RSA, scripts AGI (Asterisk Gateway Interface), base de datos astdb y archivos para el pequeño servidor HTTP para AJAM (Asynchronus Javascript Asterisk Manager). Aquí se verá una descripción de cada uno de los directorios, ya que los archivos pueden ser irrelevantes.

1) *agi-bin/:* Aquí se almacenan programas en C, PHP, Python o cualquier otro lenguaje con el que se pretenda interactuar desde Asterisk.
2) *keys/:* Directorio que contiene llaves RSA para la autenticación de llamadas con el protocolo IAX2
3) *sounds/:* Directorio con todos los sonidos que serán utilizados por aplicaciones como Playback y Background.
4) *mohmp3/:* Archivos MP3 para MusicOnHold.

Una vez que termine la compilación, y después de revisar los archivos que fueron instalados, se puede proceder a iniciar Asterisk. Varias distribuciones de GNU/Linux incluyen su propia forma de iniciar servicios. Se inicia el servicio como 'root' tecleando en línea de comandos:

asterisk –vvvvvvvvvvvvvvvvvvvvc

De esta manera se usan dos opciones para iniciarlo. Cada 'v' indica un incremento en la verbosidad, esto es, número de mensajes de notificaciones que se van a observar en la consola. Por tanto, mientras más letras 'v', más mensajes saldrán. La letra 'c' proporciona una consola en donde aparecen los mensajes de lo que sucede en la PBX y poder ejecutar comandos de Asterisk. La línea de comandos en Asterisk es sencilla de usar. Al presionar <TAB>, se podrán ver los comandos disponibles, y también completarlos. Un ejemplo de comando sería:

**CLI> sip show peers*

Este comando muestra los 'peers' (usuarios tipo peer) disponibles en el sistema. Muestra las extensiones dadas de alta en el sistema que utilizan el protocolo SIP. Existe un comando análogo para IAX2. Con esto ya está la PBX instalada. Sólo falta configurarla con las necesidades particulares de cada situación. Para ello se va a hacer uso de la aplicación de configuración de Asterisk: **FreePBX.**