*Asterisk*

*Asterisk* es una plataforma de aplicaciones (caja de herramientas) de telefonía de código abierto distribuido bajo la GPLv2. En definitiva, se trata de una aplicación de servidor para recibir y realizar un tratamiento personalizado de las llamadas telefónicas.

El proyecto fue iniciado por Mark Spencer en 1999. Mark tenía una empresa llamada Servicios de Soporte Linux y necesitaba un sistema de teléfono para ayudar a operar su negocio. Él no tenía mucho dinero para gastar en la compra de uno, por lo que termino haciendo uno propio. Como la popularidad de *Asterisk* creció, Servicios de Soporte de Linux cambió el enfoque de *Asterisk* y cambió su nombre por el de Digium, Inc.

El nombre de *Asterisk* viene del carácter \*. El objetivo del proyecto *Asterisk* es hacer todo de telefonía. A través de la consecución de este objetivo, *Asterisk* soporta ahora una larga lista de tecnologías para hacer y recibir llamadas telefónicas. Esto incluye muchos protocolos de VoIP (Voz sobre IP), así como conectividad digital a la red telefónica tradicional, o la PSTN (Public Switched Telephone Network). Esta capacidad de obtener diferentes tipos de llamadas telefónicas dentro y fuera del sistema es una de las principales fortalezas de *Asterisk*.

Una vez que se hacen las llamadas telefónicas hacia y desde un sistema *Asterisk*, hay muchas características adicionales que se pueden utilizar para personalizar el proceso de la llamada de teléfono. Algunas características son aplicaciones comunes pre-construidos, tales como correo de voz. Hay otras características más pequeñas que se pueden combinar entre sí para crear aplicaciones de voz personalizada, tales como la reproducción de un archivo de sonido, los dígitos de lectura, o el reconocimiento de voz.

***1.1. Conceptos arquitectónicos críticos***

En esta sección se discuten algunos conceptos arquitectónicos que son fundamentales para todas las partes del *Asterisk*. Estas ideas son la base de la arquitectura de *Asterisk*.

***1.1.1. Canales***

Un canal en Asterik representa una conexión entre el sistema Asterix y alguno punto final de telefonía (Figura 1.1). El ejemplo más común es cuando un teléfono hace una llamada a un sistema *Asterisk*. Esta conexión está representada por un único canal. En el código de asterik, existe un canal como una instancia de la estructura de datos ast\_channel. Este escenario de llamada podría ser una persona que llama he interactúa con el correo de voz, por ejemplo.

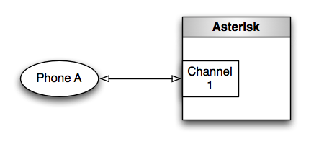


Figura 1.1: Una única llamada Representado por un canal simple

***1.1.2. Canal puente***

Tal vez un escenario de llamada más familiar sería una conexión entre dos teléfonos, en los que una persona que usa el teléfono A ha llamado a una persona en el teléfono B. En este escenario de llamada, hay dos puntos finales de telefonía conectados al sistema *Asterisk*, por lo que existen dos canales (Figura 1.2).

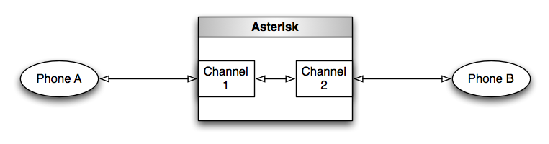


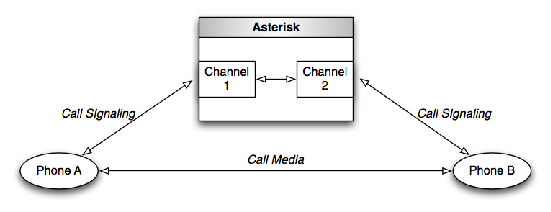
Figura 1.2: Dos tramos de llamada representado por dos canales

Cuando los canales de *Asterisk* se conectan, se le conoce como un puente canal. Puente canal es el acto de conexión de ambos canales con el propósito de pasarse los medios de comunicación. El flujo de medios es más comúnmente una secuencia de audio. Sin embargo, también puede haber un vídeo o una secuencia de texto en la llamada. Incluso en el caso en que hay más de un flujo de medios (tales como audio y vídeo), que está todavía a cargo de un único canal para cada extremo de la llamada en Asterik . En la figura 1.2, donde hay dos canales para móviles A y B , el puente es responsable de pasar los medios de comunicación procedentes de teléfono de A al teléfono B , y de manera similar , para el paso de los medios de comunicación procedentes de teléfono en teléfono B al teléfono A. Todos los flujos de medios son negociados a través de *Asterisk*. Cualquier cosa que *Asterisk* no entienda, tiene control total sobre lo que no está permitido. Esto significa que *Asterisk* puede hacer grabación, manipulación de audio, y la traducción entre diferentes tecnologías.

Cuando dos canales se unen entre sí, hay dos métodos que pueden ser utilizados para lograr esto: *puente genérico* y *puenteo nativo*.

* Un *puente genérico* es el que funciona independientemente de las tecnologías de los canales que están en uso. Se pasa todo el audio y la señalización a través de las interfaces de canal abstractos *Asterisk* . Aunque este es el método de puente más flexible, también es el menos eficiente debido a los niveles de abstracción necesario para llevar a cabo la tarea. Figura 1.2 ilustra un puente genérico.
* Un *puente nativo* es un método específico de tecnología de conexión de canales unidos. Si dos canales están conectados a Asterik usando la misma tecnología de transporte de los medios de comunicación, puede haber una manera de conectarlos que es más eficiente que ir a través de las capas de abstracción en Asterik. Por ejemplo, si el hardware especializado se utiliza para la conexión a la red telefónica, puede ser posible superar los canales en el hardware de manera que los medios de comunicación no tiene que fluir hacia arriba a través de la aplicación en absoluto. En el caso de algunos protocolos de VoIP, es posible tener puntos finales envían sus flujos de medios entre sí directamente, de tal manera que sólo la información de señalización de llamada continúa fluyendo a través del servidor.

La decisión entre *puente genérico* y *puente nativo* se realiza mediante la comparación de los dos canales, cuando es el momento de superar las capas de abstracción. Si los dos canales de ellos indican que respaldan el mismo método de puente nativo entonces se utilizará. De lo contrario, se utilizara el método de puente genérico. Para determinar si dos canales soportan el mismo método de puente nativo, se utiliza una comparación simple puntero a función C. Ciertamente no es el método más elegante, pero aún no han afectado a los casos en que esto no era suficiente para nuestras necesidades. Proporcionar una función de puente nativo para un canal se discute en más detalle en la Sección 1.2. La figura 1.3 ilustra un ejemplo de un puente nativo.



***1.1.3. Marcos.***

La comunicación dentro del código de *Asterisk* durante una llamada se realiza mediante el uso de marcos, que son instancias de la estructura de datos ast\_frame. Los marcos pueden ser marcos de los medios de comunicación o marcos de señalización. Durante una llamada telefónica básica, una corriente de tramas de medios que contienen audio se pasa a través del sistema. Los Marcos de señalización se utilizan para enviar mensajes sobre eventos de la señalización de llamada (call singnalling), como un dígito está presionado, las llamadas que se pone en espera o una llamada se colgó.

La lista de tipos de marcos disponibles se define estáticamente. Los marcos están marcados con un tipo numéricamente codificado y subtipo. Una lista completa se puede encontrar en el código fuente include/*Asterisk*/frame.h algunos ejemplos son:

* VOICE: Estos marcos tienen una porción de una secuencia de audio.
* VIDEO: Estos marcos tienen una porción de una secuencia de vídeo.
* MODEM: Contiene la codificación utilizada para los datos de este marco, como T.38 para el envío de un fax a través de IP. El uso principal de este tipo de trama es para el manejo de un FAX. Es importante que las tramas de datos puedan dejar completamente intacto de modo que la señal puede ser descodificada con éxito en el otro extremo. Esto es diferente de las tramas de audio, porque en ese caso, es aceptable para transcodificar en otros códecs de audio para ahorrar ancho de banda en el costo de la calidad de audio.
* CONTROL: mensaje de la señal de llamada que indica los marcos. Estas tramas se utilizan para indicar eventos de señalización de llamada. Estos eventos incluyen un teléfono que se responde y colgó, puso en espera , etc
* DTMF\_BEGIN: El dígito que hay que empezar. Este marco se envía cuando una persona que llama presiona un tecla DTMF en su teléfono.
* DTMF\_END: El dígito que acaba de terminar. Este marco se envía cuando una persona que llama deja de presionar una tecla DTMF en su teléfono.

***1.2.*** *Asterisk* ***Abstracciones de componentes***

*Asterisk* es una aplicación altamente modular. Hay una aplicación central que se construye a partir de la fuente en el main / directorio principal del árbol de código fuente. Sin embargo, no es muy útil por sí mismo. La aplicación principal actúa principalmente como un registro de módulo. También cuenta con un código que sabe cómo conectar todas las interfaces abstractas conjunto para hacer que las llamadas telefónicas funcionen. Las implementaciones concretas de estas interfaces son registrados por los módulos que se pueden cargar en tiempo de ejecución.

Por defecto, todos los módulos que se encuentran en un directorio de módulos de *Asterisk* predefinido en el sistema de archivos se cargan cuando se inicia la aplicación principal. Este enfoque fue elegido por su simplicidad. Sin embargo, existe un archivo de configuración que puede ser actualizado para especificar exactamente qué módulos se carguen y en qué orden cargarlos. Esto hace que la configuración sea un poco más compleja, pero ofrece la posibilidad de especificar que los módulos que no son necesarios no se carguen. *El principal beneficio es la reducción de la huella de memoria de la aplicación*. Sin embargo, hay algunos b*eneficios de seguridad*, también. Lo mejor es no cargar un módulo que acepta conexiones a través de una red si no es realmente necesario.

Cuando se carga el módulo, registra todas sus implementaciones de las captaciones de los componentes con la aplicación central *Asterisk*. Hay muchos tipos de interfaces que los módulos pueden implementar y registrar con el núcleo de *Asterisk*. Se permite un módulo para registrar el mayor número de estas interfaces diferentes como quisiera. En general, la funcionalidad relacionada se agrupa en un solo módulo.

***1.2.1. Drivers de canal***

La interfaz del controlador de canal de *Asterisk* es la más compleja y la más importante. La API de canal *Asterisk* proporciona el protocolo de la abstracción de telefonía que permite a todas las demás características de *Asterisk* trabajar de forma independiente del protocolo de telefonía en uso. Este componente se encarga de la traducción entre la abstracción canal de *Asterisk* y los detalles de la tecnología de telefonía que implementa.

La definición de la interfaz del controlador de canal de *Asterisk* se llama la interfaz ast\_channel\_tech. Se define un conjunto de métodos que se deben implementar por un conductor del canal. El primer método que un controlador de canal debe implementar es un método de fábrica ast\_channel, que es el método requester en ast\_channel\_tech. Cuando se crea un canal de *Asterisk* , ya sea para una llamada telefónica entrante o saliente , la aplicación de ast\_channel\_tech asociado con el tipo de canal que se necesita es responsable de la creación de instancias y la inicialización de la ast\_channel para esa llamada .

Una vez que un ast\_channel ha sido creado, contendrá una referencia a la ast\_channel\_tech que lo creó. Hay muchas otras operaciones que deben ser manejados de una manera específica dependiendo la tecnología. Cuando estas operaciones se deben realizar en un ast\_channel, el manejo de la operación difiere del método apropiado de ast\_channel\_tech. La Figura 1.2 muestra dos canales en *Asterisk*. Figura 1.4 se extiende en este para mostrar dos canales en puente y cómo las implementaciones de tecnología de canal encaja en la imagen.

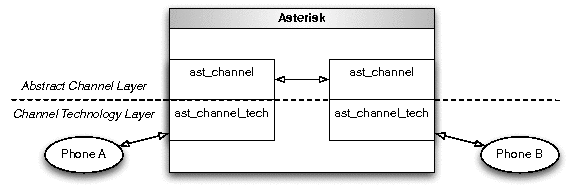


Figura 1.4: Capa de canal de la tecnología y capas de canales abstractos.

Los métodos más importantes en ast\_channel\_tech son:

requester: Es la respuesta al controlador de canal cuando este solicita la creación de instancias de un objeto ast\_channel e inicializareste tipo de canal.

call: Respuesta que se utiliza para iniciar una llamada saliente al punto final representado por un ast\_channel.

answer: Se llama cuando *Asterisk* decide que debe responder a la llamada de entrada asociada a este ast\_channel.

hangup: Esto se llama cuando el sistema ha determinado que la llamada debe ser colgado. El conductor de canal debe comunicarse con el punto final que la llamada ha terminado de una manera específica del protocolo.

indicate: Una vez que la llamada se ha terminado, hay una serie de otros eventos que puedan ocurrir esa necesidad de ser señalado a un punto final . Por ejemplo , si el dispositivo se pone en espera , esta devolución de llamada se llama para indicar esa condición. Puede haber un método específico protocolo de indicar que la llamada ha estado en espera , o el controlador de canal puede simplemente iniciar la reproducción de música en espera en el dispositivo.

send\_digit\_begin: Esta función es llamada para indicar el comienzo de un dígitos ( DTMF ) de ser enviado a este dispositivo .

send\_digit\_end: Esta función se llama para indicar el final de un dígito (DTMF ) que se envían a este dispositivo.

read: Esta función es llamada por el núcleo de *Asterisk* para leer de nuevo un ast\_frame de este extremo. Un ast\_framees una abstracción en el asterisco que se utiliza para encapsular los medios de comunicación ( tales como de audio o de vídeo ) , así como para señalar eventos .

write : Esta función se utiliza para enviar un ast\_frame a este dispositivo. El conductor del canal se llevará a los datos y paquetes como apropiado para el protocolo de telefonía que lleva a cabo y pasar a lo largo de la estación.

bridge: Este es el puente nativo de respuesta para este tipo de canal. Como se discutió antes , de puenteo nativo es cuando un conductor del canal es capaz de implementar un método más eficiente de puente para dos canales del mismo tipo en lugar de tener toda la señalización y el flujo de medios de comunicación a través de capas de abstracción innecesarias adicionales . Esto es muy importante por razones de rendimiento.

Una vez que la llamada ha terminado, el código de control de canal abstracto que vive en el núcleo de *Asterisk* invocará la devolución de llamada colgar ast\_channel\_tech y luego destruir el objeto ast\_channel.

***1.2.2. Aplicaciones*** *Dialplan* ***(Lógica de mercado)***

El dialplain es el corazón de arterisk.

Los Administradores de *Asterisk* configuran el enrutamiento de llamadas utilizando el *dialplan* de *Asterisk*, que reside en el archivo /etc/*Asterisk*/extensions.conf. El diaplain se compone de una serie de reglas de llamadas denominas extensiones. Cuando entra una llamada de teléfono en el sistema, el número marcado se utiliza para encontrar la extensión en el dialplain que se debe utilizar para el procesamiento de la llamada. La extensión incluye una lista de aplicaciones *dialplan* que se ejecutarán en el canal. Las aplicaciones disponibles para su ejecución en el *dialplan* se mantienen en un registro de aplicación. Este registro se rellena en tiempo de ejecución como se cargan los módulos.

*Asterisk* tiene casi doscientas aplicaciones incluidas. Las aplicaciones pueden utilizar cualquiera de las API internas de *Asterisk* para interactuar con el canal. Algunas aplicaciones hacen una sola tarea, tales como la reproducción, la cual se reproduce un archivo de sonido a la persona que llama. Otras aplicaciones realizan un gran número de operaciones, tales como la aplicación de correo de voz.

Usando el *dialplan* de *Asterisk*, varias aplicaciones se pueden utilizar conjuntamente para personalizar la gestión de llamadas. Si se necesita más amplia personalización más allá de lo posible en el idioma *dialplan* proporcionado, hay interfaces disponibles scripting que permite la manipulación para personalizar el uso de cualquier lenguaje de programación llamada. Incluso cuando se utilizan estas interfaces de scripting con otro lenguaje de programación, aplicaciones *dialplan* todavía invocan para interactuar con el canal.

Antes de entrar en un ejemplo, vamos a echar un vistazo a la sintaxis de un *dialplan* de *Asterisk* que se encarga de las llamadas al número 1234. Tenga en cuenta que la elección de 1234 aquí es arbitraria. Invoca tres aplicaciones *dialplan*. En primer lugar, responde a la llamada. A continuación, se reproduce un archivo de sonido. Por último, se cuelga la llamada.

Define the rules for what happens when someone dials 1234.

exten => 1234,1,Answer()

same => n,Playback(demo-congrats)

same => n,Hangup()

La palabra clave exten se utiliza para definir la extensión. En el lado derecho de la línea exten, el 1234 significa que estamos definiendo las reglas para cuando alguien llama a 1234. El próximo 1 significa que este es el primer paso que se toma cuando se marca ese número. Por último, Answer indica al sistema contestar la llamada. Las siguientes dos líneas que comienzan con la palabra clave same son reglas para la última extensión que se ha especificado, que en este caso es 1234. La n es para decir que este es el siguiente paso a tomar. El último elemento de esas líneas especifica qué acción tomar.

***1.2.3. Funciones*** *Dialplan*

La mayoría de las aplicaciones de *dialplan* toman una cadena de argumentos. Mientras que algunos valores pueden ser codificados, se utilizan las variables en los lugares donde el comportamiento tiene que ser más dinámico. El siguiente ejemplo muestra un fragmento *dialplan* que establece una variable y luego imprime su valor a la interfaz de línea de comandos de *Asterisk* usando la aplicación Verbose.

exten => 1234,1,Set(MY\_VARIABLE=foo)

same => n,Verbose(MY\_VARIABLE is ${MY\_VARIABLE})

Las funciones *dialplan* se invocan usando la misma sintaxis que el ejemplo anterior. Los módulos de *Asterisk* son capaces de registrar funciones *dialplan* que pueden recuperar algo de información y de retorno al *dialplan*. Por otra parte, estas funciones *dialplan* pueden recibir datos del *dialplan* y actuar en consecuencia. Como regla general, mientras que las funciones *dialplan* pueden establecer o recuperar los metadatos de canal, no hacen ninguna señalización o procesamiento de medios. Eso queda como el trabajo de las aplicaciones *dialplan*.

El siguiente ejemplo muestra el uso de una función *dialplan*. En primer lugar, se imprime el identificador de llamadas del canal actual a la interfaz de línea de comandos de *Asterisk*. Luego, cambia el identificador de llamadas utilizando la aplicación \_ID. En este ejemplo, \_Verbose y \_ID son aplicaciones, y \_CALLERID es una función.

exten => 1234,1,Verbose(The current CallerID is ${CALLERID(num)})

same => n,Set(CALLERID(num)=<256>555-1212)

Se necesita una función *dialplan* aquí en lugar de sólo una variable simple ya que la información se almacena en el identificador de llamadas estructuras de datos en la instancia de ast\_channel. El código de función *dialplan* sabe cómo establecer y recuperar los valores de estas estructuras de datos.

***1.3. Temas***

*Asterisk* es una aplicación muy fuertemente multiproceso. Utiliza la API del hilo POSIX para manejar los hilos y los servicios relacionados, tales como el bloqueo. Todo el código de *Asterisk* que interactúa con hilos lo hace yendo a través de un conjunto de envolturas usadas para propósitos de depuración. La mayoría de las discusiones en *Asterisk* pueden clasificarse ya sea como un subproceso de supervisión de red, o un hilo de canal (a veces también se conoce como un hilo PBX, ya que su propósito principal es ejecutar el PBX para un canal ) .

***1.3.1. Monitor de red de Tema***

Existen hilos del monitor de red en todos los principales impulsores de canal en *Asterisk*. Ellos son responsables de supervisar cualquier red se conectan a ( ya sea una red IP, la RTB , etc ) y controlar las llamadas entrantes u otros tipos de peticiones entrantes. Ellos se encargan de los pasos iniciales de establecimiento de conexión, tales como la autenticación y validación de número marcado. Una vez que la configuración de la llamada se ha completado, los hilos de monitorización creará una instancia de un canal de *Asterisk* ( ast\_channel ) , y comenzar un hilo de canal para manejar la llamada para el resto de su vida.

***1.3.2. Temas Canal***

Como se señaló anteriormente, un canal es un concepto fundamental en *Asterisk*. Los canales son ya sea entrante o saliente. Se crea un canal de entrada cuando se recibe una llamada al sistema *Asterisk*. Estos canales son los que ejecutan el *dialplan* de *Asterisk* . Se crea un hilo para cada canal de entrada que se ejecuta el plan de marcado. Estos hilos se denominan hilos de canal.

Aplicaciones *Dialplan* siempre se ejecutan en el contexto de un canal de hilo. Funciones *Dialplan* casi siempre lo hacen, también. Es posible leer y escribir funciones *dialplan* de una interfaz asíncrona como el asterisco CLI. Sin embargo, todavía es siempre el hilo canal que es el propietario de la estructura de datos ast\_channel y controla la duración del objeto.

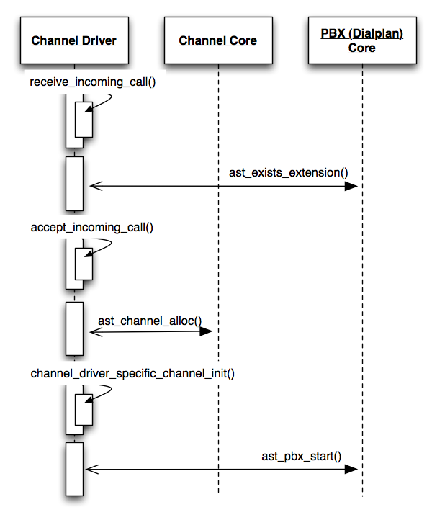
***1.4. Escenarios de Llamada.***

Algunos escenarios de llamadas comunes se dividen para demostrar cómo los componentes de *Asterisk* operan en conjunto para procesar las llamadas telefónicas.

**1.4.1. Comprobación del correo de voz**

Un ejemplo de un escenario de llamada es cuando alguien llama a la central telefónica para comprobar su correo de voz. El primer componente principal involucrado en este escenario es el impulsor de canal. El conductor del canal será responsable de manejar la petición de llamada entrante del teléfono, que se producirá en detector de hilo del conductor del canal. Dependiendo de la tecnología de telefonía que se utiliza para entregar la llamada al sistema, puede haber algún tipo de negociación requerida para establecer la llamada. Otro paso de la creación de la llamada es determinar el destinatario de la llamada. Esto generalmente viene especificado por el número marcado por el llamante. Sin embargo, en algunos casos, no hay un número específico disponible ya que la tecnología utilizada para entregar la llamada no admite especificar el número marcado. Un ejemplo de esto sería una llamada entrante en una línea telefónica analógica.

Si el controlador del canal verifica que la configuración de *Asterisk* tiene extensiones definidas en el plan de marcado ( la configuración de enrutamiento de llamadas) para el número marcado , entonces se asignará un objeto de canal *Asterisk* ( ast\_channel ) y crear un hilo de canal. El hilo de canal tiene la responsabilidad primordial de manejar el resto de la llamada (Figura 1.5).



El bucle principal que el canal maneja es la ejecución dialplain. Se va a las reglas definidas para la extensión marcada y ejecuta las medidas que se han definido. El siguiente es un ejemplo de extensión expresada en la sintaxis plan de marcado extensions.conf. Esta ampliación responde a la llamada y ejecuta la aplicación \_VoicemailMain cuando alguien marca \* 123. Esta aplicación es lo que un usuario podría llamar para poder comprobar los mensajes dejados en su buzón de correo.

exten => \*123,1,Answer()

same => n,VoicemailMain()

Cuando el hilo de canal ejecuta la aplicación \_Answer, *Asterisk* contesta la llamada entrante. Responder a una llamada requiere la tecnología de procesamiento específico, lo que además de un manejo respuesta genérica, la devolución de llamada de respuesta de la estructura ast\_channel\_tech asociado está llamado a manejar de contestar la llamada. Esto puede suponer el envío de un paquete especial sobre una red IP, tomando una línea analógica descolgado, etc

***1.4.2. Llamada puenteada***

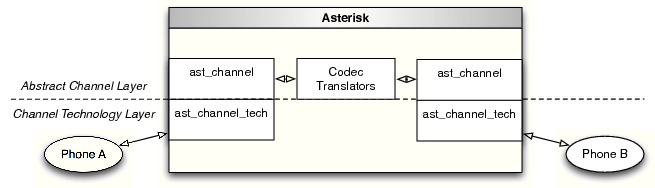
Otro escenario llamada muy común en *Asterisk* es una llamada en puente entre dos canales. Este es el escenario cuando uno llama a otro teléfono a través del sistema. El proceso de establecimiento de llamada inicial es idéntica a la del ejemplo anterior. La diferencia en el manejo comienza cuando la llamada se ha establecido y el hilo de canal comienza a ejecutar el plan de marcado.

El siguiente plan de marcado es un ejemplo sencillo que resulta en una llamada en puente. Utilizando esta extensión, cuando un teléfono marca 1234, el *dialplan* ejecutarán la aplicación Dial, que es la aplicación principal que se utiliza para iniciar una llamada saliente.

exten => 1234,1,Dial(SIP/bob)

El argumento especificado para la aplicación \_Dial dice que el sistema debe hacer una llamada saliente al dispositivo denominado \_sip / bob . La porción SIP de este argumento especifica que el protocolo SIP se debe utilizar para entregar la llamada. \_bob será interpretado por el controlador del canal que implementa el protocolo SIP , chan\_sip . Suponiendo que el controlador del canal ha sido adecuadamente configurado con una cuenta llamada \_bob , será saber cómo llegar a teléfono de Bob.

La aplicación \_Dial pedirá al núcleo de *Asterisk* para asignar un nuevo canal de *Asterisk* utilizando el identificador SIP / bob . El núcleo solicitará que el controlador del canal SIP realizar la inicialización específica tecnología. El conductor del canal también iniciará el proceso de hacer una llamada a teléfono. Como la solicitud de fondos , pasará eventos de nuevo en el núcleo de *Asterisk*, que será recibida por la aplicación de marcación . Estos eventos pueden incluir una respuesta que la llamada ha sido contestada , el destino está ocupado , la red está congestionada , la llamada fue rechazado por alguna razón, o un número de otras posibles respuestas . En el caso ideal, se responderá la llamada. El hecho de que la llamada ha sido contestada se propaga de nuevo al canal de entrada. *Asterisk* no responderá a la parte de la llamada que entró en el sistema hasta que se responda a la llamada saliente. Una vez que ambos canales son contestadas , el puente de los canales comienza (Figura 1.7).



Durante un puente canal, eventos de audio y señalización de un canal se pasan a la otra hasta que se produzca algún evento que hace que el puente hasta el final, como por ejemplo uno de los lados de la llamada colgando el teléfono. El diagrama de secuencia de la Figura 1.8 muestra las operaciones fundamentales que se realizan por una trama de audio durante una llamada en puente.

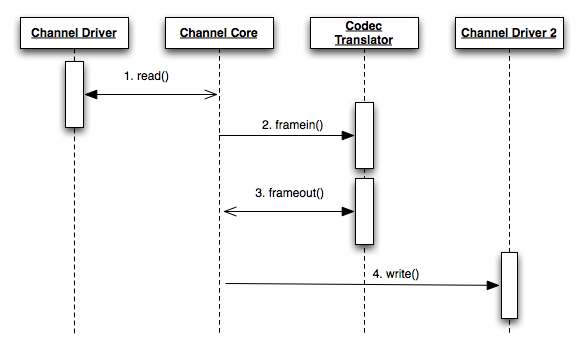


Figura 1.8: Diagrama de secuencia para el procesamiento de tramas de audio durante un Puente

Una vez que la llamada se hace, el proceso de colgado es muy similar al ejemplo anterior. La diferencia principal aquí es que hay dos canales implicados. El procesamiento de colgar específica tecnología de canal se ejecutará para ambos canales antes de que el hilo de canal deja de funcionar.

***1.5. Comentarios Finales***

La arquitectura de *Asterisk* hoy en día tiene más de 10 años. Sin embargo, los conceptos fundamentales de canales y gestión de llamadas flexibles utilizando el *dialplan* de *Asterisk* siguen apoyando el desarrollo de los sistemas de telefonía complejos en una industria que está en constante evolución. Una de las áreas que la arquitectura de *Asterisk* no trata muy bien es la ampliación de un sistema a través de múltiples servidores. La comunidad de desarrollo de *Asterisk* está desarrollando actualmente un proyecto del compañero llamado *Asterisk* SCF (marco de Comunicaciones escalables) que está destinada a hacer frente a estas preocupaciones de escalabilidad. En los próximos años, esperamos ver *Asterisk*, junto con *Asterisk* SCF, haciendose cargo de una parte significativa del mercado de la telefonía, incluidas las instalaciones mucho más grandes.