Introducción a la VoIP a través de Elastix

Memoria del Proyecto

José Álvaro Perera Domínguez
UOC 2013

Índice

1. Introducción

El presente documento corresponde al Proyecto Fin de Carrera (en adelante PFC) titulado "Introducción a la VoIP a través de Elastix".
Se comienza describiendo los objetivos del proyecto, así como las causas que han motivado su elección. Se continúa con el alcance, una planificación detallada de las tareas a realizar y una relación de los entregables. A continuación, se desarrolla el contenido del PFC.

2. Objetivos del proyecto

Los objetivos del proyecto son los siguientes:

- Conocer el estado de la telefonía antes de la popularización de VoIP.
- Estudiar la tecnología VoIP.
- Conocer Asterisk, montando para ello una PBX que use VoIP, usando la distribución Linux Elastix.

Se persigue con este trabajo, principalmente, que el alumno adquiera las competencias necesarias para poder aportar opiniones con fundamento en parte de su actividad laboral. Se espera que al adquirir un punto de vista lo suficientemente amplio y fundamentado para ver como encaja la tecnología IP en el mundo de la telefonía, se adquiera la competencia de ver que ventajas e inconvenientes supone cada posible solución a un problema, y elegir la mejor.

3. Alcance

En el PFC se comenzará con una descripción de la telefonía, comenzando por sus orígenes hasta el presente, sin incluir de momento VoIP. En este estudio se hará un repaso de las distintas tecnologías, de sus implantaciones comerciales, así como de la terminología existente en este mundo.

Se continuará con la descripción de la tecnología que compone VoIP, de los problemas y ventajas que tiene, así como de su situación actual en el mercado de las telecomunicaciones.

Una vez adquirida la base necesaria, se implementará una PBX que use tecnología VoIP mediante el software Asterisk. Para ello, se instalará y configurará la distribución de Linux denominada Elastix. Finalmente, se harán pruebas de la PBX mediante softphones, ya sea desde sistemas operativos Windows o Android.

4. Planificación

4.1. Tareas principales del proyecto

A partir de una estimación inicial se han identificado las siguientes tareas principales y su duración:

Elaboración del plan de trabajo.

- Objetivo: seleccionar el PFC, definir su objetivo, alcance y elaborar la planificación

temporal.

- Duración: 14 días.
- Producto: plan de trabajo del proyecto.

Introducción a la telefonía.

- Objetivo: conocer la tecnología telefónica "tradicional", sin incluir VoIP.
- Duración: 9 días.
- Producto: el estudio realizado formará parte de la PEC 2.

Estudio de VoIP

- Objetivo: conocer la tecnología VoIP.
- Duración: 21 días.
- Producto: una parte del estudio realizado formará parte de la PEC 2, y otra parte de la PEC 3.

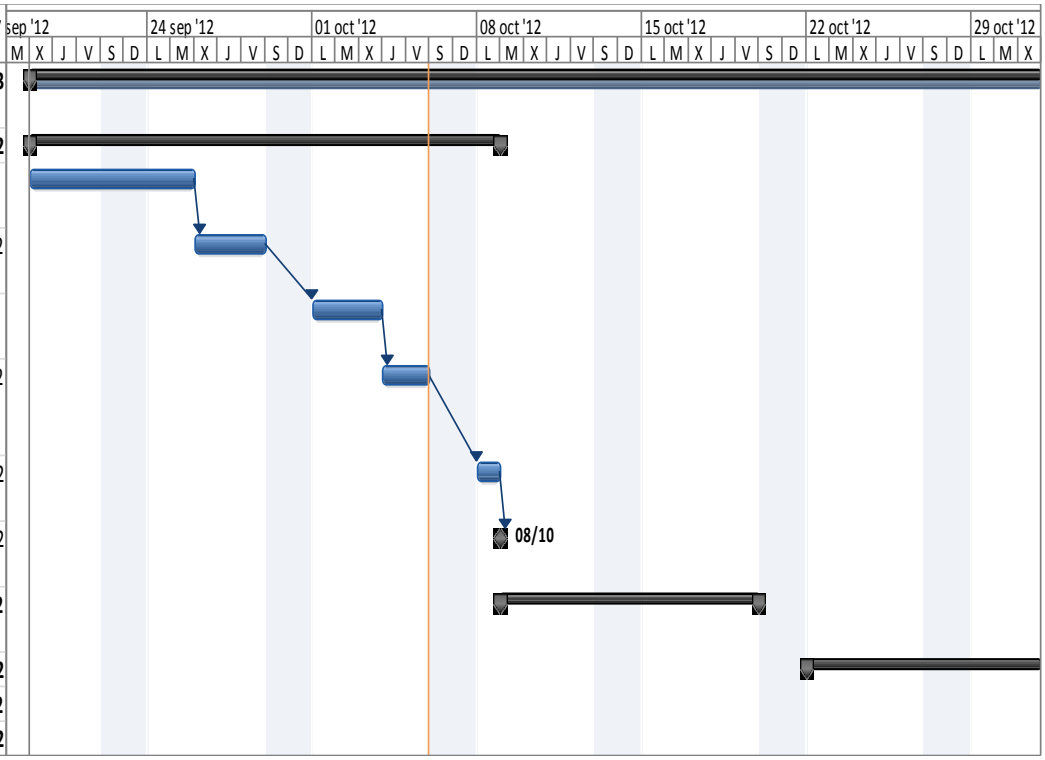
Instalación y configuración de Asterisk en Elastix

- Objetivo: conocer el software Asterisk y montar una PBX que use VoIP sobre una distribución Linux Elastix.
- Duración: 14 días.
- Producto: Asterisk funcionando como central PBX con varias extensiones IP configuradas.

4.2. Calendario de trabajo

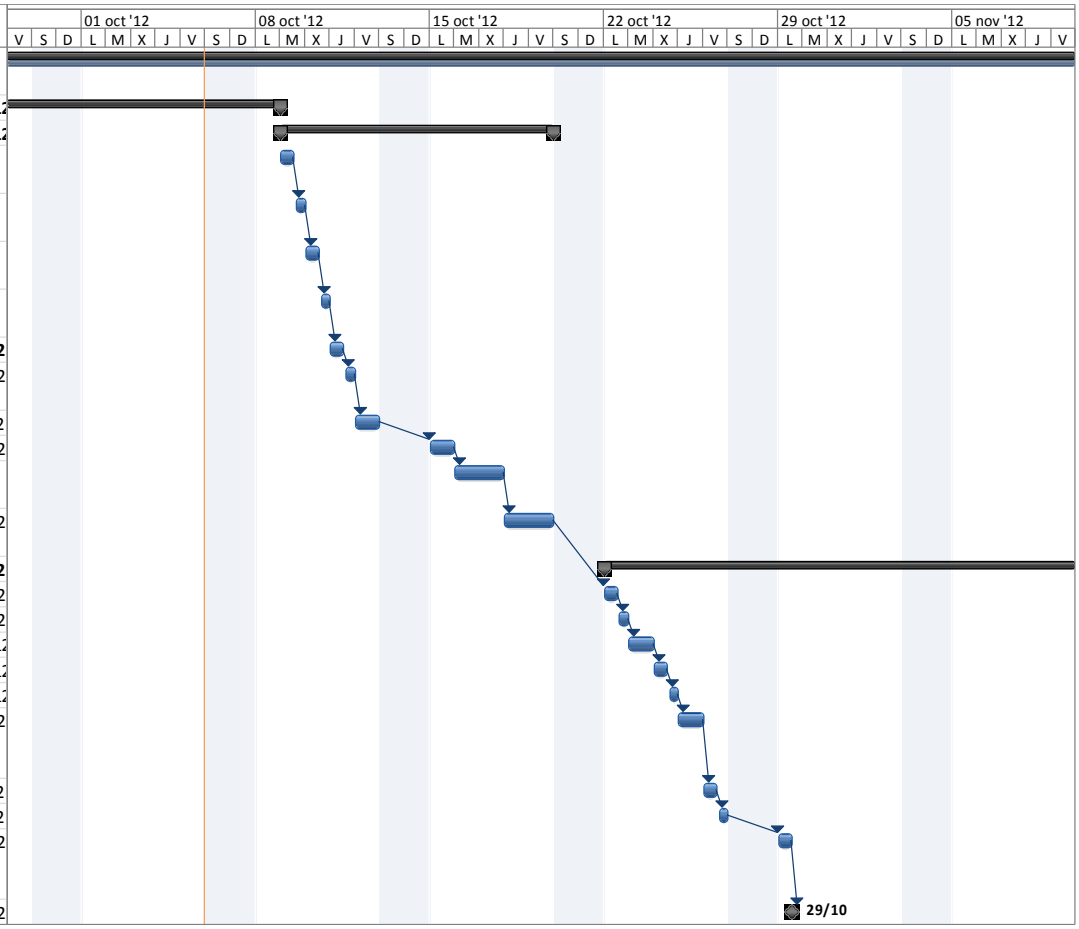
El diagrama de Gantt con las tareas a realizar en la PEC 1 es:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Redacción de la memoria del PFC**	**79 días**	**mié 19/09/12**	**lun 07/01/13**
2	**Plan de trabajo**	**14 días**	**mié 19/09/12**	**lun 08/10/12**
3	Selección del proyecto	5 días	mié 19/09/12	mar 25/09/12
4	Definición de alcance y objetivos	3 días	mié 26/09/12	vie 28/09/12
5	Definición de las tareas a realizar	3 días	lun 01/10/12	mié 03/10/12
6	Elaboración de la planificación temporal del PFC	2 días	jue 04/10/12	vie 05/10/12
7	Elaboración de documentos	1 día	lun 08/10/12	lun 08/10/12
8	Entrega de la PEC1: Plan de trabajo	0 días	lun 08/10/12	lun 08/10/12
9	**Introducción a la telefonía**	**9 días**	**mar 09/10/12**	**vie 19/10/12**
20	**Tecnología VoIP**	**20,5 días**	**lun 22/10/12**	**lun 19/11/12**
37	**Asterisk**	**14 días**	**lun 19/11/12**	**vie 07/12/12**
48	**Entrega final**	**14 días**	**vie 07/12/12**	**jue 27/12/12**



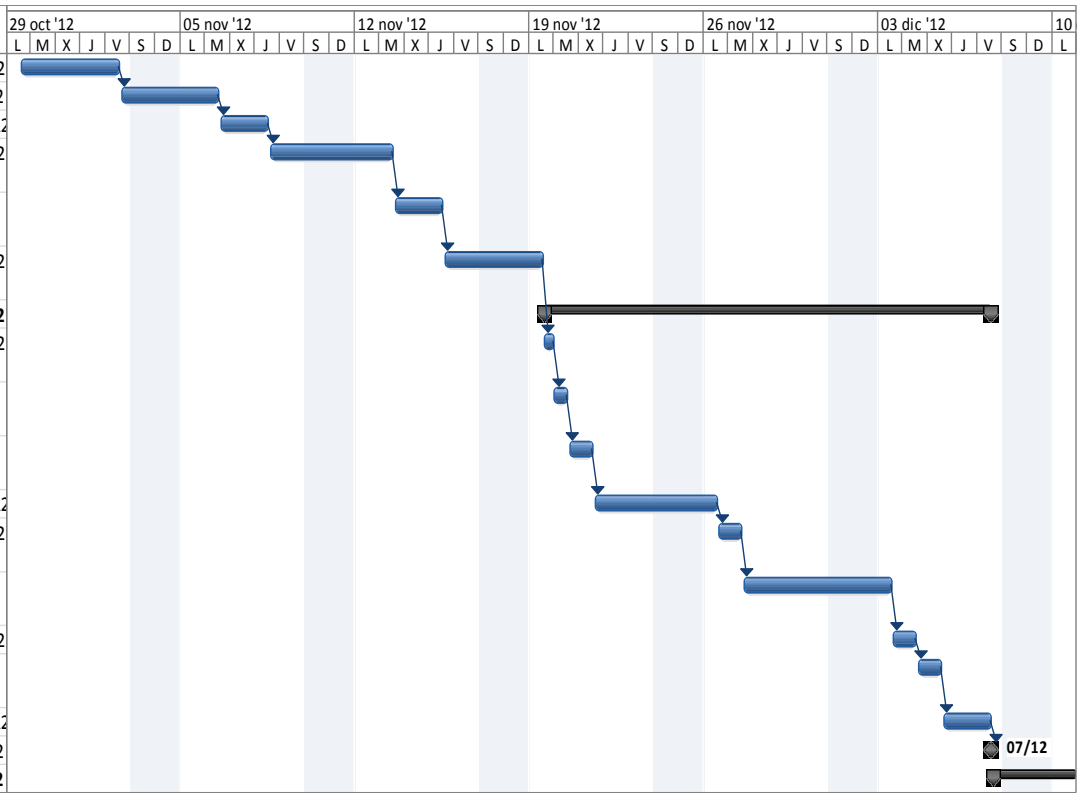
Para la PEC 2:

Id	Nombre de tarea	Duración	Comienzo
1	**Redacción de la memoria del PFC**	**79 días**	**mié 19/09/12**
2	**Plan de trabajo**	**14 días**	**mié 19/09/12**
9	**Introducción a la telefonía**	**9 días**	**mar 09/10/12**
10	Historia de la telefonía	0,5 días	**mar 09/10/12**
11	Transmisión de la voz humana	0,5 días	**mar 09/10/12**
12	Digitalización de la voz	0,5 días	**mié 10/10/12**
13	Redes orientadas a circuitos o paquetes	0,5 días	**mié 10/10/12**
14	Definición de PSTN	0,5 días	**jue 11/10/12**
15	Tecnología analógica y digital	0,5 días	jue 11/10/12
16	Circuitos analógicos	1 día	vie 12/10/12
17	Circuitos digitales	1 día	lun 15/10/12
18	Telefonía empresarial	2 días	mar 16/10/12
19	Terminología y conceptos	2 días	jue 18/10/12
20	**Tecnología VoIP**	**20,5 días**	**lun 22/10/12**
21	Modelo OSI	0,5 días	lun 22/10/12
22	Protocolo IP	0,5 días	lun 22/10/12
23	Protocolos de transport	1 día	mar 23/10/12
24	Latencia	0,5 días	mié 24/10/12
25	Jitter	0,5 días	mié 24/10/12
26	Modulación por impulsos codificados (PCM)	1 día	jue 25/10/12
27	Compresión de la voz	0,5 días	vie 26/10/12
28	Pérdida de paquetes	0,5 días	vie 26/10/12
29	Comparativa establecimiento de llamada PSTN/VoIP	0,5 días	lun 29/10/12
30	Entrega PEC 2	0 días	lun 29/10/12



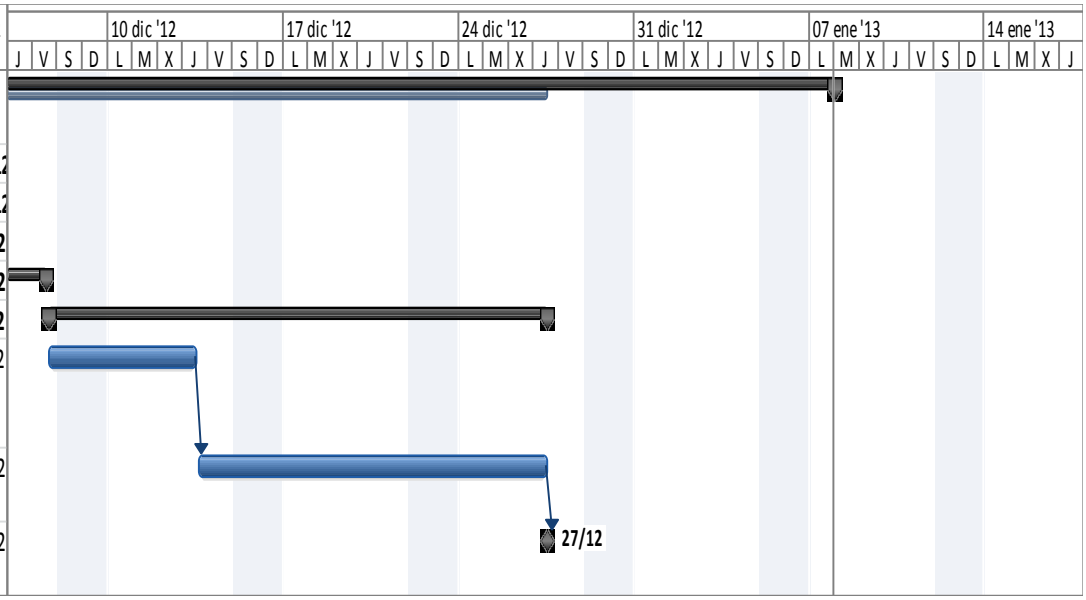
Para la PEC 3:

Id	Nombre de tarea	Duración	Comienzo
31	Quality of Service	4 días	lun 29/10/12
32	Servicios de facturación	2 días	vie 02/11/12
33	Seguridad	2 días	mar 06/11/12
34	Protocolos de señalización IP	3 días	jue 08/11/12
35	Interconexión entre PSTN y VoIP	2 días	mar 13/11/12
36	Aplicaciones y servicios sobre VoIP	2 días	jue 15/11/12
37	**Asterisk**	**14 días**	**lun 19/11/12**
38	Introducción a Asterisk	0,5 días	lun 19/11/12
39	Introducción a Elastix	0,5 días	mar 20/11/12
40	Instalación de Elastix	1 día	mar 20/11/12
41	Configuración	3 días	mié 21/11/12
42	Instalación de softphones	1 día	lun 26/11/12
43	Funcionamiento interno de freePB X	4 días	mar 27/11/12
44	Calidad de voz	1 día	lun 03/12/12
45	Diagnóstico de problemas	1 día	mar 04/12/12
46	Seguridad	2 días	mié 05/12/12
47	Entrega PEC 3	0 días	vie 07/12/12
48	**Entrega final**	**14 días**	**vie 07/12/12**



Para la entrega final:

Id	Nombre de tarea	Duración	Comienzo
1	**Redacción de la memoria del PFC**	**79 días**	**mié 19/09/12**
2	**Plan de trabajo**	**14 días**	**mié 19/09/12**
9	**Introducción a la telefonía**	**9 días**	**mar 09/10/12**
20	**Tecnología VoIP**	**20,5 días**	**lun 22/10/12**
37	**Asterisk**	**14 días**	**lun 19/11/12**
48	**Entrega final**	**14 días**	**vie 07/12/12**
49	Presentación OpenOffice.org Impress	4 días	vie 07/12/12
50	Corrección de errores y revisión	10 días	jue 13/12/12
51	Entrega Memoria Final	0 días	jue 27/12/12



5. Entregables

5.1. PEC 1

En la primera entrega se elabora el presente Plan de Proyecto, en el que se describe lo que se va a hacer, cuales son los objetivos, el alcance, una primera aproximación a la descomposición en tareas y la planificación temporal de dichas tareas.

5.2. PEC 2

La segunda entrega consiste en el estudio de la telefonía tradicional, así como de parte de la tecnología VoIP.

5.3. PEC 3

La tercera entrega consiste en el resto del estudio de VoIP, así como la documentación asociada a la instalación de la centralita Asterisk, su configuración con VoIP y las pruebas de funcionamiento desde varios softphones.

5.4. Memoria final

En la última entrega se incluirá toda la documentación generada en las anteriores entregas, así como las correcciones o revisiones que se hayan realizado. También se incluye una presentación del proyecto para un rápido conocimiento del PFC.

6. Bibliografía

LANDIVAR, Edgar. Comunicaciones Unificadas con Elastix. Volumen 1. 2009.

LANDIVAR, Edgar. Comunicaciones Unificadas con Elastix. Volumen 2. 2009.

DAVIDSON; PETERS; BHATIA; KALIDINDI; MUKHERJEE. Voice over IP Fundamentals, Second Edition. 2006.

SHARIF, BEN. Elastix without Tears. 2008.

7. Introducción a la telefonía

7.1. Breve historia de la telefonía

La idea de poder comunicarse remotamente mediante algún dispositivo que trasmitiese la voz seguro que fue objeto de muchas cavilaciones desde tiempos inmemoriales por parte de muchos inventores, pero no fue hasta los inicios del siglo diecinueve cuando, con la invención de la electricidad, empezó a parecer factible.
En 1849, un médico italiano llamado Antonio Meucci hizo una demostración en la Habana de un dispositivo capaz de transmitir la voz. El mismo Meucci, en 1854, volvió a mostrar su invento de Nueva York. Como ya se había inventado el telégrafo, el objetivo en estos años era

inventar un "telégrafo parlante".
En 1860, el alemán Johann Philipp Reis construye un aparato capaz de transmitir la voz basándose en una idea, nunca llevada a la práctica, de Charles Bourseul. El dispositivo de Reis fue mejorándose, llegando a transmitir a más de 100 metros de distancia.
En 1871, Meucci intentó patentar su invento mediante un aviso de patente, pero debido a la falta de recursos económicos no pudo terminar el trámite, expirando su aviso de patente a los pocos años.
Un año después de expirar el aviso de patente realizado por Meucci, Alexander Graham Bell, escocés residente en los Estados Unidos, patenta un dispositivo similar, siendo el primero en conseguir la patente. Curiosamente, otro inventor llamado Elisha Gray intentó patentar un invento muy parecido sólo unas pocas horas después de Bell, desatándose mucha polémica y dando lugar a una disputa legal que finalmente ganó Bell.
Además de habilidad como inventor, Bell tenía sobre todo talento como empresario, ya que convirtió el teléfono en un negocio rentable, por lo que suyo es el mérito de convertir la idea en algo útil para la sociedad.
No obstante, parece que el tema no estaba del todo cerrado, ya que el 11 de junio de 2002, el Congreso de los Estados Unidos aprobó la resolución 269, que reconocía que el teléfono lo había inventado Antonio Meucci, y no Alexander Graham Bell.
La telefonía ha ido evolucionando a la misma velocidad que el resto de avances tecnológicos. Ya no se requiere que una operadora conecte manualmente los cables para realizar la conmutación. En 1891 se inventó un teléfono que permitía marcar directamente y conmutar automáticamente hacia el otro abonado.
Aunque al principio la compañía Bell fue prácticamente un monopolio, debido a la concesión de sus patentes, cuando éstas expiraron fueron surgiendo cientos de pequeñas compañías que empezaron a hacer la competencia a Bell, y a extender el negocio de la telefonía a gran velocidad.
A finales de la segunda guerra mundial el número de personas con teléfono ya era del orden de millones.

7.2. Transmisión de la voz humana

Para transmitir la voz humana fue necesario esperar a poder dominar la electricidad, ya que la voz humana se atenúa rápidamente, no pudiendo escuchar una conversación a unos pocos metros. Usando ondas eléctricas, transmidas a través de un conductor metálico como el cobre, viaja a una velocidad muy elevada, del orden de la velocidad de la luz, por lo que desde el punto de vista del planeta la comunicación es casi instantánea.
A partir de este conocimiento, ya a mediados del siglo XIX se comprendía que el proceso adecuado era convertir ondas acústicas en eléctricas para transmitirlas a grandes distancias. Una cuestión clave, por tanto, era inventar un dispositivo que hiciera esta labor, que hoy en día conocemos como micrófono.

7.3. El micrófono

Una característica importante de la voz humana es que las cuerdas vocales modulan la voz en un amplio espectro de frecuencias que van de graves a agudos en un rango aproximado de 20Hz a 20kHz.
No obstante, no es necesario que un micrófono puede capturar todo ese rango de frecuencias

completo, ya que para obtener un resultado entendible basta con transmitir el intervalo de 400Hz a 4kHz.
Como dijimos, el micrófono fue un elemento clave en la invención del teléfono, ya que convierte las ondas acústicas a ondas eléctricas. Hay muchos tipo de micrófonos, realizando cada uno la conversión basándose en diferentes principios. Uno muy usado por mucho tiempo fue el de carbón. Consistía en una cápsula llena de granitos de carbón entre dos placas metálicas. Una de las placas vibraba según las ligeras presiones de las ondas de voz, por lo que la resistencia eléctrica de la cápsula variaba y se generaba la señal eléctrica correspondiente.

Ilustración 1: Típico micrófono de carbón extraído de teléfono de disco

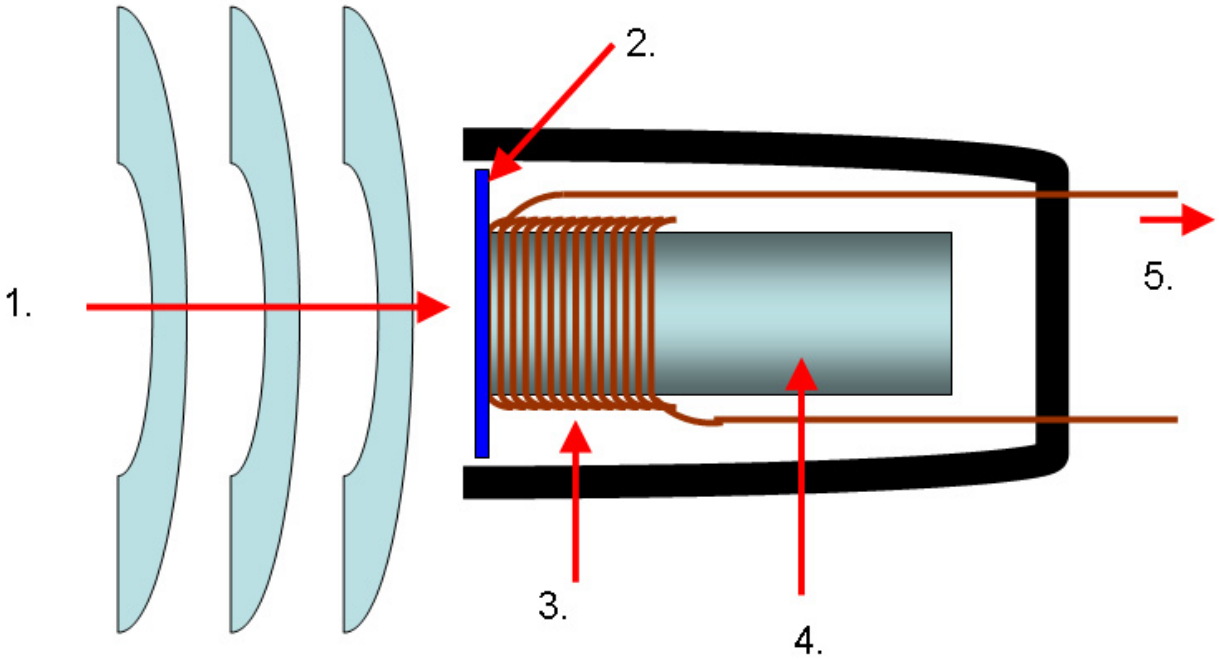


Ilustración 2: Diagrama esquemático de micrófono electro-magnético

Otro tipo de micrófono muy común en la actualidad es el dinámico o electro-magnético. Consiste en una bobina de hilo de cobre enrollada sobre un núcleo de material ferromagnético. Este núcleo se encuentra sujetado a un diafragma que vibra con la presión de las ondas de voz, por lo que se induce una corriente eléctrica en la bobina, según la voz.
En la figura 2 podemos observar algunos componentes del micrófono electromagnético, mientras reaccionan ante un frente de ondas acústicas:

1. Ondas de voz
2. Diafragma
3. Bobina
4. Núcleo ferromagnético
5. Corriente inducida

7.4. Digitalización de la voz

La digitalización, o conversión analógica-digital (CAD), consiste en la conversión de señales analógicas en digitales. Mediante este proceso, se facilitan tratamientos como la codificación, compresión, etc., y se consigue también que aguante mejor el ruido y las interferencias que las señales analógicas.
En la figura 3 vemos las etapas que componen un CAD:

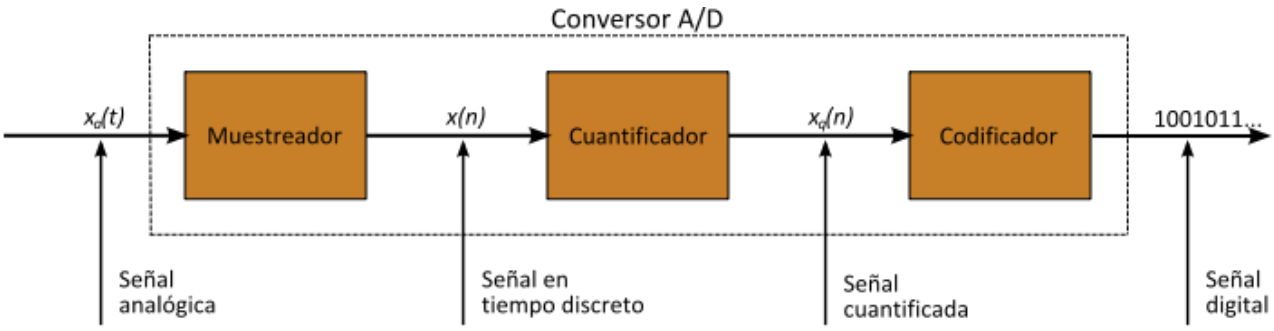


Ilustración 3: Procesos de la conversión A/D

Las ventajas de la señal digital son:

1. Se reconstruye y amplifica muy facilmente.
2. Se detectan y corrigen errores de transmisión fácilmente.
3. Operaciones de procesamiento de la señal se realizan fácilmente.
4. Se puede regenerar infinitas veces sin pérdida de calidad.

Los cuatro procesos que intervienen en la conversión analógica-digital son:

1. Muestreo: el muestreo (sampling) consiste en tomar muestras periódicas de la amplitud de onda. La velocidad con que se toma esta muestra, es decir, el número de muestras por segundo, es lo que se conoce como frecuencia de muestreo.
2. Retención (hold): las muestras tomadas han de ser retenidas por un circuito de retención (hold), el tiempo suficiente para permitir evaluar su nivel (cuantificación).
3. Cuantificación: consiste en asignar un único nivel de salida a un margen de valor de entrada de una señal analizada.
4. Codificación: consiste en traducir los valores obtenidos durante la cuantificación al código binario.

En 1928, Harry Nyquist formuló un teorema que decía que, como mínimo, se necesita como frecuencia de muestreo el doble del ancho de banda de la señal de entrada. La expresión es la siguiente:

$$f_m \geq 2\ BW_s$$

Sabiendo que la voz humana tiene un rango de frecuencias entre 400 a 4kHz, según el teorema de Nyquist haría falta muestrear a 8 kHz. Posteriormente veremos que esa es la frecuencia que usan la mayoría de los codecs.

7.5. Redes orientadas a circuitos o paquetes

Las redes telefónicas tradicionalmente han sido orientadas a circiutos, que significa que antes de iniciar una comunicación es necesario establecer un circuito entre los dos extremos. Una vez establecido, no puede ser usado por otros. Cuando finalice la conversación, se liberan los recursos usados, que volverán a estar disponibles para otros abonados.

Una de las ventajas de las redes orientadas a circuitos es que el retardo es constante. Una vez establecido, al ser un camino físico, se puede contar con que la calidad de la conexión no cambie. Sin embargo, este tipo de redes es caro, ya que se necesita un circiuto dedicado para cada conversación activa. Este tipo de redes es el usado por las compañías telefónicas, y, salvando las distancias por los avances tecnológicos, es el mismo tipo que usó Bell en sus inicios.

No hay que confundir las redes de circuitos con redes para transmitir señales analógicas, ya que aunque las redes de circuitos tradicionalmente se han usado para transmitir este tipo de señales, nada impide que se transmitan también señales digitales.
En el otro lado están las redes de paquetes, que es una red por la que viajan por un mismo medio paquetes correspondientes a diferentes flujos de información. Para conseguir esto, fragmenta el tráfico de cada flujo en paquetes y los envía de manera intercalada. Posteriormente, en el destino los paquetes se vuelve a unir para obtener de nuevo el mensaje original.

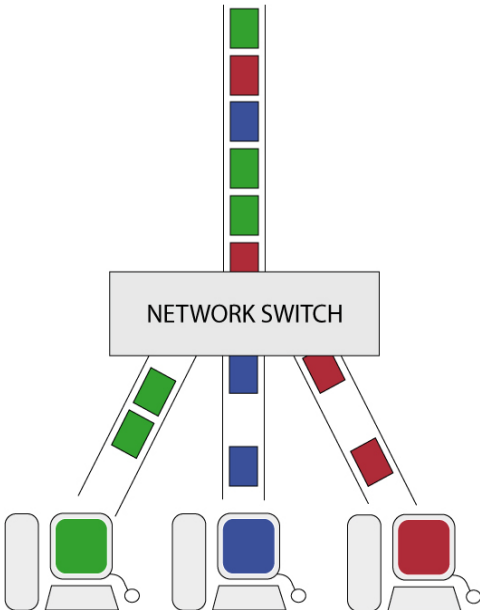


Ilustración 4: Simulación gráfica de envío de paquetes en una red de paquetes

Internet, como toda red IP, es un ejemplo de red de este tipo, ya que por una misma conexión pueden llegar distintos flujos de información, como video-conferencia a la vez que se envía un correo o navegamos por la Web. A diferencia de las redes orientadas a circuitos, el ancho de banda no es fijo, ya que según el tráfico que haya en un momento dado se necesitará más o menos capacidad. Además, cada uno de los paquetes puede seguir un camino distinto para llegar al mismo destino, por lo que hay que lidiar con el desorden y la pérdida de los mismos. Este tipo de redes se han hecho muy populares debido a que optimizan el uso de las redes, y abre la puerta además a muchos servicios relacionados.

7.6. Definición de PSTN

Public Switched Telephone Network (PSTN), o Red Telefónica Conmutada (RTC), se define como el conjunto de medios de transmisión y conmutación necesarios para comunicar dos terminales mediante un circuito físico establecido para dicha comunicación, y que se elimina una vez finalizada. En otras palabras, se trata de la red telefónica clásica.
Los componentes de toda red RTC son:

- Terminal de abonado y línea telefónica de abonado (también llamada bucle local).
- Centrales de conmutación de circuitos.
- Sistema de transmisión.

- Sistema de Señalización.

Las características básicas de una PSTN son:

- Ofrece a cada usuario un circuito para señales analógicas con una banda base de 4KHz para cada conversación.
- Única red con cobertura nacional, donde la red se va ramificando progresivamente en subredes que llevan cada vez menor tráfico.
- Capacidad de interconexión con las redes móviles.
- El costo para el usuario por la ocupación del circuito varía en función de la duración de la conexión y la distancia entre los extremos.
- La comunicación la hace valiéndose de centrales de conmutación, a las que se conectan los medios de transmisión.
  Los medios de transmisión entre centrales de conmutación se conocen como troncales, y en la actualidad transportan principalmente señales digitales sincronizadas. En cambio, los medios de transmisión entre los equipos domiciliarios y las centrales continúan siendo pares de cobre, y se les sigue llamando líneas de abonado. Hay otras formas de acceder al domicilio, como enlaces inalámbricos, fibra óptica, RDSI, xDSL, aunque no se las suele considerar parte de la telefonía básica.

7.7. Circuitos analógicos

Un circuito analógico es un par de líneas de cobre, que unen físicamente la central de conmutación más cercana con la ubicación del abonado. En esta línea se transmite la señal eléctrica correspondiente a la voz de los dos extremos de la comunicación de manera analógica. En el mismo circuito se reciben o envían la señalización necesaria para establecer, mantener y finalizar una llamada.
Aunque durante muchos años los circuitos analógicos han monopolizado las redes telefónicas, desde la irrupción de las comunicaciones digitales el circuito analógico se ha visto reducido prácticamente a "la última milla", que es como se denomina al tramo de la comunicación que corresponde del abonado a su central de conmutación. De esta forma, aunque la señal salga y entre del terminal de abonado en formato analógico, en algún momento dicha señal es convertida en digital.
Como en muchas zonas rurales no había electricidad, muchas redes telefónicas proveen voltaje para alimentar los teléfonos. La central de conmutación genera 48 voltios de corriente para el funcionamiento del terminal.

7.7.1. Señalización analógica

Es necesario contar con un mecanismo de señalización para intercambiar información entre el abonado y su central. Por ejemplo, para indicarnos cuando nos llama alguien, o para indicar nosotros cuando finalizamos la llamada.
Básicamente hay tres formas de señalización analógica: loop start, ground start y kewlstart. Es importante saber, si se configura una central telefónica, cual es es el método de señalización adecuado, ya que de lo contrario podemos encontrarnos con que el teléfono se comporta de forma rara.
La diferencia entre loop start y ground start radica en la forma en que el teléfono pide tono de marcado. En ground start requiere tono de marcado poniendo a tierra uno de los conductores

de la línea telefónica, mientras que en loop start se realiza realiza una conexión entre ambos. Kewlstart es una evolución de loop start.
Se explica a continuación la señalización para los eventos más comunes, colgado, descolgado, marcación, conmutación, sonado y conversación, usando señalización loop start.

7.7.1.1. Colgado

Cuando el teléfono esta colgado, la central de conmutación proporciona una corriente continua de 48 voltios, pero como el circuito en el teléfono está abierto, no fluye corriente.

7.7.1.2. Descolgado

Cuando el usuario descuelga el teléfono, cierra el circuito del par de cables de la línea telefónica a través de una resistencia. En el momento que la central detecta esto, envía el tono de marcado para que el abonado sepa que ya puede marcar el número. El tono de marcado cambia según el país, siendo el más común en Europa una onda de 425 Hz.

7.7.1.3. Marcacion

El proceso de marcación puede ser por pulsos o tonos, aunque los pulsos casi no se usan, pero era el mecanismo habitual en los teléfonos de disco.
Los tonos consisten en pares de frecuencias que se asocian con dígitos, y que son transmitidos a la central, que los traduce a números.
A veces también hay que enviar dígitos en medio de una conversación, para lo que se ideó el estandard Dual-Tone Multi-Frequency (DTMF). Cada tono corresponde a dos tonos mezclados enviados a la vez por la línea telefónica.

	1209 Hz	**1336 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Tabla 1: Tonos DTMF

7.7.1.4. Conmutación

Cuando la central reciba el número marcado, determinará si el abonado depende directamente de ella o no. En caso de no ser un abonado local, se pondrá en contacto con otro conmutador para que se haga cargo de la entrega de la llamada.

7.7.1.5. Timbrado o ringado

Cuando la central localiza al abonado destino de la llamada, le enviará la señal de timbrado o ring. Consiste en una onda senoidal de 20 Hz y 90 V de amplitud.
Como complemento a la señal de ring, se envía también al emisor de la llamada la señal de ring-back, para que sepa que se está realizando la llamada. Dicha señal está compuesta por dos ondas senoidales de 440Hz y 480Hz.

Si el destinatario estuviese ocupado, se devolvería el tono ocupado en lugar del de ring-back. Este tono consiste en dos ondas senoidales superpuestas, de 480Hz y 620Hz.

7.7.1.6. Conversación

Cuando el destinatario de la llamada responda el teléfono, cerrará el circuito, señal que la llegará a la central de conmutación, que completará al fin el circuito de conexión.

7.7.2. El teléfono analógico

Como ya se comentó anteriormente, fue la invención del teléfono analógico lo que marcó el desarrollo del negocio de la telefonía. Además, y a pesar del tiempo transcurrido, sigue siendo el tipo de teléfono más común.
El teléfono analógico, en su forma más básica, está formado por un número de componentes muy reducido:

- Auricular.
- Micrófono.
- Interruptor para colgado/descolgado.
- Convertidor de dos a cuatro hilos: también llamado bobina híbrida, es necesario para separar la señal de audio del emisor y del receptor, ya que sólo hay un par de cables para dos ondas. Este componente históricamente ha sido origen de problemas como el eco en las líneas mal acopladas.
- Marcador.
- Campana o dispositivo de timbrado.

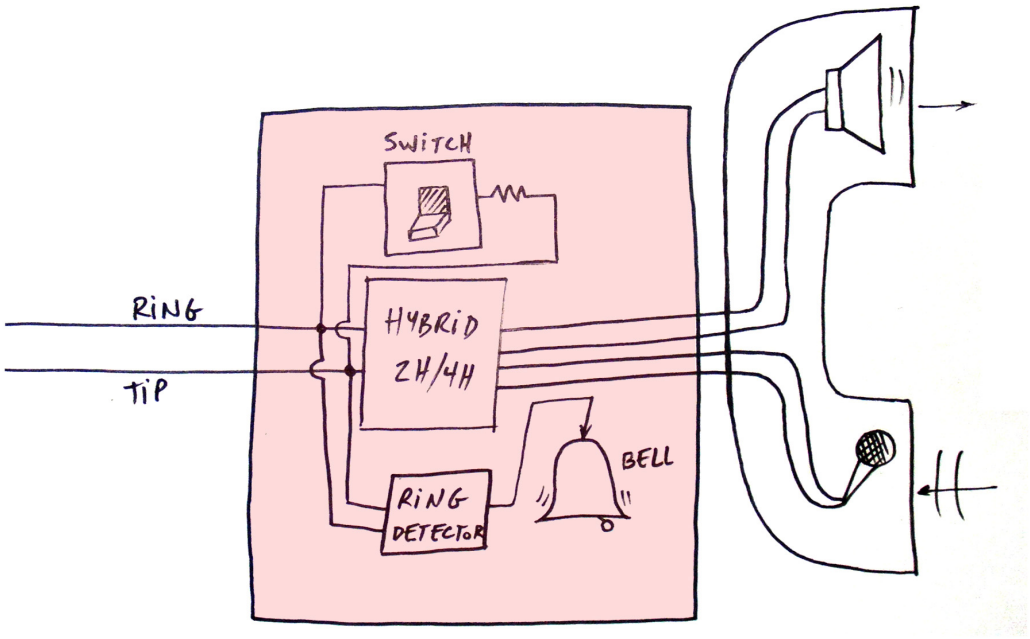


Ilustración 5: Diagrama de componentes de un teléfono

7.8. Circuitos digitales

La RTB, o PSTN, también puede funcionar con circuitos digitales. Entre otras ventajas, estos circuitos permiten multiplexar más de una línea por un mismo medio físico, por lo que puede resultar interesante si se necesitan varias líneas telefónicas. Algunos de los formatos en los que se comercializa son:

- Base DS-0: es un canal digital de 64Kbit/s. Se utiliza como unidad estandar a partir de la cual definir circuitos de mayor capacidad.
- Circuitos T-carrier y E-carrier: los T-carrier fueron diseñados por los laboratorio Bell hace más de cincuenta años, siendo los E-carrier su equivalente europeo. El más conocido es el T1, y su correspondiente E1. El T1 está compuesto por 24 DS-0, y el E1 está compuesto por 32 DS-0. El T1 tiene un ancho de banda de 1544 Mbit/s, y el E1 de 2048 Mbit/s. Posteriormente a los T1 y E1, fueron saliendo múltiplos suyos como T2, T3, T4 y T5.
- SONET y Circuitos Opticos: SONET (Synchronous optical networking) fue desarrollado con el objetivo de contar con una nomenclatura similar a las T-carrier pero usando la tecnología de fibra óptica. SONET utiliza múltiplos de T3 para sus anchos de banda y su

circuito base es el llamado OC-1. Luego del OC-1 tenemos los OC-3, OC-12, OC-24, OC-48, entre otros.

7.8.1. Protocolos de señalización digital

Al igual que en la telefonía analógica, la telefonía digital también necesita de señalización para los eventos normales de un canal de comunicaciones, como "sonando", "descolgando", etc., así como otra información, como identificación de llamante, etc.
Los protocolos de señalización digital se dividen en dos tipos, CAS (Channel Associated Signaling) y CCS (Common Channel Signaling).

7.8.1.1. Señalización asociada al canal (CAS)

En los protocolos CAS la señalización se transmite por el mismo canal en el que viaja la información. Uno de los protocolos CAS más populares es robbed-it, usado en los circuitos T1 y E1. Dicho protocolo coge, de cada seis frames, el octavo bit, el menos significativo, y lo reemplaza por información de señalización, perdiéndose el bit original.
Esto es posible hacerlo debido en conversaciones de voz debido a que el oído no es muy sensible a esta pérdida, al ser el bit menos significativo, pero al transportar datos la pérdida de un bit provoca que la calidad de la transmisión se degrade sensiblemente.

7.8.1.2. Señalización de Canal Común (CSS)

En los protocolos CCS se señaliza por un canal separado, lo que provoca que el ancho de banda útil se reduzca, por lo que se han adoptado de forma general los protocolos CAS.
Un ejemplo de señalización CSS es la red ISDN (Integrated Services Digital Network), o como se comercializó en España, RDSI, que nos permite transmitir voz y datos simultáneamente sobre pares telefónicos de cobre con calidad superior a las líneas telefónicas analógicas.
RDSI salió con objeto de popularizar las líneas digitales a los usuarios finales, ofreciendo una amplia gama de servicios integrados. RDSI define dos tipo de conexiones, primarias y básicas:

- BRI: Basic Rate Interface
- PRI: Primary Rate Interface

BRI, o acceso básico, estaba orientado a hogares y pequeñas empresas. Un acceso básico consta de dos canales útiles, llamados canales B, de 64 kBit/s cada uno, más uno de señalización de 16 kBit/s, llamado canal D. Este tipo de acceso tuvo muy poca acogida en los Estados Unidos, aunque en Europa sí ha disfrutado de algo más de éxito.
PRI, o acceso primario, es una opción para organizaciones con mayores requisitos, siendo muy popular tanto en Europa como en Estados Unidos. Se utilizan sobre circuitos T-carrier o E-carrier.

8. Tecnología VoIP

El uso del protocolo IP como protocolo de transporte es la clave de la mayoría de los beneficios de la voz sobre IP (VoIP). Para enteder realmente estos beneficios, hay que entender lo que IP significa, como se comporta y como es un paquete IP.
No obstante, para entender lo que IP puede hacer y de que manera, hay que conocer el modelo OSI (Open Systems Interconnection), y como se aplica a IP.

8.1. Modelo OSI

La International Organization for Standardization (ISO) desarrolló el modelo OSI a finales de los años ochenta. OSI es ahora mismo el estandar de facto para el desarrollo de protocolos de comunicación entre dispositivos en red. Aunque no todos los protocolos siguen este modelo, la mayoría de los nuevos protocolos usan el enfoque de aproximación por capas.

El modelo OSI enfoca el problema de la comunicación entre máquinas dividiéndolo en siete niveles. Cada nivel se preocupa de comunicarse sólo con su correspondiente nivel en la otra máquina (ver Ilustración 6). Esto significa que el nivel 5 no tiene que preocuparse por las funciones del nivel 1, que se encarga del nivel físico.

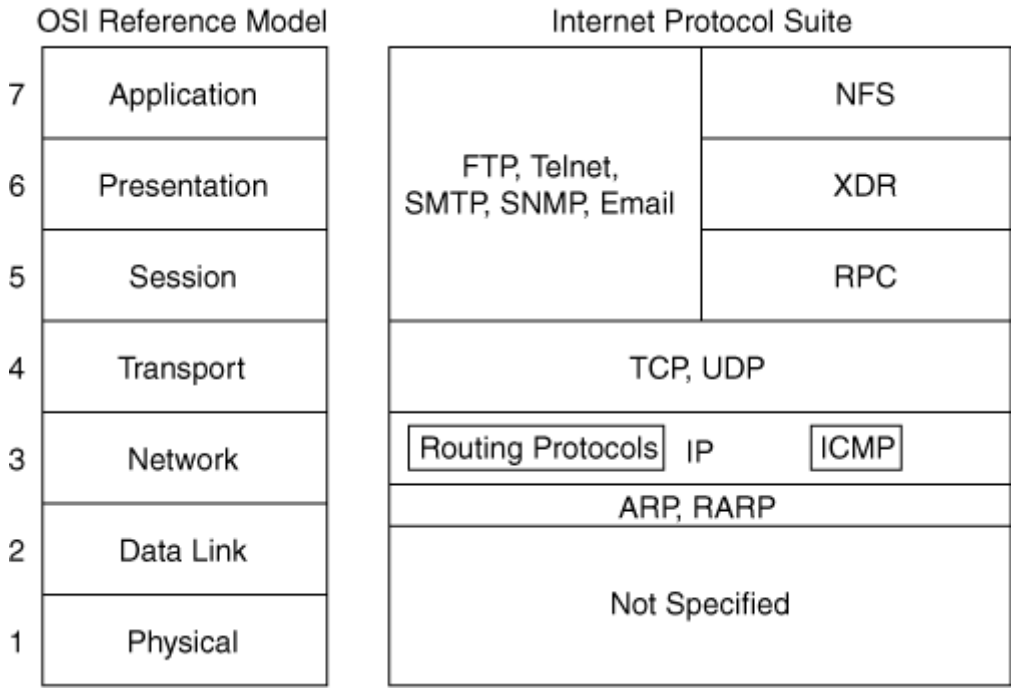


Ilustración 6: Modelo de referencia OSI

Además, cada capa del modelo OSI provee servicios a su capa inmediatamente superior (la capa 5 a la capa 6, la capa 6 a la capa 7, etc) y pide ciertos servicios a su capa inmediatamente inferior (la capa 5 a la 4, la capa 4 a la 3, etc.)

Esta aproximación por niveles permite a cada nivel manejar pequeñas piezas de información, hacer pequeños cambios a los datos, y añadir la información correspondiente al nivel antes de pasarlo al siguiente nivel.

A continuación se describirán los siete niveles (aplicación, presentación, sesión, transporte,

red, enlace de datos y físico). Entender estos niveles permitirá enteder como funciona el enrutamiento IP, y como es transportado a través de los medios físicos de la capa 1.

La correspondencia del conjunto de protocolos de Internet con los niveles OSI se muestra en la Ilustración 6.

8.1.1. El nivel de Aplicación

Es el nivel que ven los usuarios, y corresponde a las aplicaciones finales. Algunas de las muchísimas aplicaciones posibles son:

- E-mail
- Navegación Web
- P2P

8.1.2. El nivel de Presentación

El nivel de Presentación garantiza que la información enviada por el nivel de Aplicación de un sistema llegue al otro extremo. Si es necesario, el nivel de Presentación traduce entre múltiples formatos de datos, usando un formato de representación común.

El nivel de Presentación también negocia aspectos de la transferencia de datos entre extremos, como por ejemplo:

- Encriptación
- Compresión
- Uso de ASCII o EBCDIC

8.1.3. El nivel de Sesión

Como su nombre sugiere, el nivel de Sesión establece, gestiona y finaliza sesiones entre aplicaciones, además de gestionar su intercambio de datos. Las sesiones de diálogos se dan entre dos o más entidades de presentación.

8.1.4. El nivel de Transporte

El nivel de Transporte es responsable de asegurar el transporte de datos fiable en una conexión entre redes. Esto se consigue usando control de flujo, chequeo de errores, reconocimientos (acknowledgments) de extremo a extremo, retransmisiones y secuencias de datos.

Algunos protocolos de nivel de Tranporte, como Transmission Control Protocol (TCP), tienen mecanismos para manejar la congestión. TCP ajusta sus tiempos de retransmisión, por ejemplo, cuando hay congestión o pérdida de paquetes en una red.

8.1.5. El nivel de Red

El nivel de Red proporciona un direccionamiento lógico que permite que dos sistemas

separados en diferentes redes puedan establecer un camino para comunicarse. La capa de Red es el nivel en el que residen los protocolos de enrutado.

Actualmente, el direccionamiento IP es el usado en Internet. Protocolos de enrutado como Enhanced Interior Gateway Routing Protocol (Enhanced IGRP, or EIGRP), Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Intermediary System to Intermediary System (IS-IS), y muchos más, son los usados para determinar la ruta óptima entre dos redes.

Las funciones principales de este nivel incluyen las siguientes:

- Formateo de paquetes, direccionamiento de redes y hosts, resolución de direcciones, enrutado.
- Creación y mantenimiento de tablas de enrutado.

8.1.6. El nivel de Enlace de Datos

El nivel de Enlace de Datos proporciona un transporte fiable a través de un enlace físico. Este nivel tiene su propio esquema de direccionamiento, y tiene relación directa con la conectividad física. Los switches Ethernet conmutan tráfico basándose las direcciones de este nivel, direcciones MAC.

8.1.7. El nivel físico

El nivel físico se refiere a leer y escribir 0s y 1s en un medio físico, por ejemplo con cambios de voltaje. Las especificaciones más usadas en este nivel son:

- EIA/TIA-232: se suele usar para conectar un ordenador a un modem.
- RS-449: especificación usada para comunicaciones síncronas de largo alcance. El conector físico usa 37 pins y tiene bastante más alcance que EIA/TIA-232.
- 802.3: uno de los estándares físicos más usados, Ethernet. Actualmente, su velocidad varía desde 10Mbps a 1000Mbps.

8.2. Protocolo IP

El protocolo IP es un protocolo sin conexión que reside en el nivel 3 (el nivel de red), lo que significa que no tiene mecanismos de fiabilidad, control de flujo, secuenciamiento o reconocimiento. Son otros protocolos de la capa superior (capa 4, sesión), como TCP, los que añaden estas características.

Dada su posición en el modelo OSI, IP no tiene que lidiar con los problemas típicos del nivel de enlace de datos a los que se enfrenta Ethernet, ATM, Frame Relay, Token Ring.

Un paquete IP puede ser direccionado de tres maneras:

- Unicast: es la más simple, la dirección identifica a un host específico, por lo que sólo un nodo debe recibir el paquete.
- Broadcast: los paquetes son enviados a todos los usuarios en una red local, ya que los broadcast no son enrutados por los routers.

- Multicast: los paquetes usan un direccionamiento especial que habilita a un grupo de usuarios de diferentes subredes a recibir el mismo tráfico.

Los paquetes unicast, broadcast y multicast tienen cada uno diferentes usos. Los paquetes unicast permiten a dos estaciones comunicarse entre sí, independientemente de su localización física. Los paquetes broadcast se usan para comunicarse con toda una subred simultáneamente. Los paquetes multicast permiten que aplicaciones como la videoconferencia tengan un transmisor y múltiples receptores.

La Ilustración 7 muestra la estructura de un paquete IP:

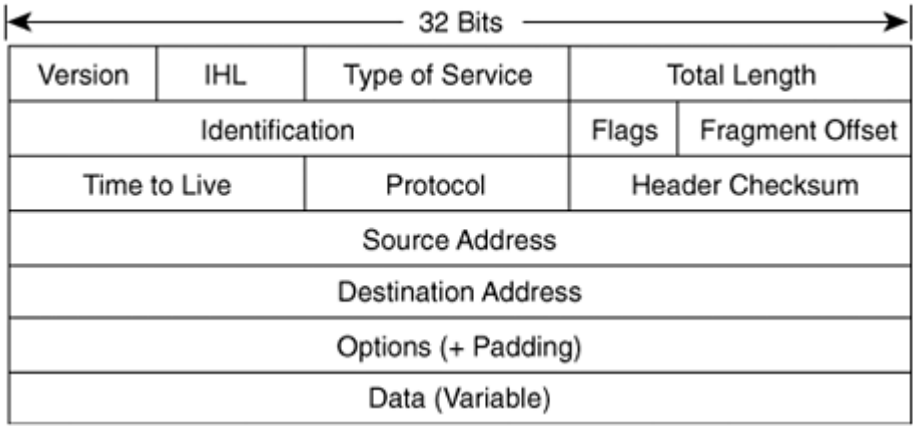


Ilustración 7: Estructura de un paquete IP

Los campos son los siguientes:

- Version: indica si se va a usar IPv4 o IPv6.
- IP header length (IHL): longitud de la cabecera del datagrama, en palabras de 32 bits,
- Type of service: especifica como debe manejar este paquete un protocolo de nivel superior, por ejemplo para calidad de servicio (QoS).
- Total length: longitud en bytes del paquete IP.
- Identification: número entero que identifica al datagrama.
- Flags: campo de 3 bits, de los cuales 2 de ellos controlan la fragmentación, no usándose el tercero.
- Time To Live: contador que va reduciéndose a cero, momento en el que se descarta el datagrama.
- Protocol: indica que protocolo de nivel superior recibirá los paquetes.
- Header checksum: verifica que la cabecera no está corrupta.
- Source address: dirección de origen.
- Destination address: dirección de destino.
- Options: opciones varias de IP, como la seguridad.
- Data: datos de aplicación más datos para los protocolos de capas superiores.

8.3. Protocolos de transporte

Hay múltiples protocolos de transporte, aunque los más usados son TCP y UDP (User Datagram Protocol). TCP y UDP tienen diferentes características, a usar según la aplicación. Si la fiabilidad es más importante que el retraso, se debe usar TCP/IP para garantizar la entrega.

8.3.1. TCP

TCP proporciona a los protocolos del nivel superior un servicio de comunicación full-duplex, con reconocimiento (ACK) y control de flujo. Mueve los datos según un stream de bytes continuo ordenados por un número de secuencia. Soporta numerosas conversaciones en el nivel superior. El número de puerto en una cabecera TCP identifica una conversación de nivel superior.

Para maximizar el rendimiento, TCP permite que cada nodo envíe varios paquetes antes de recibir el ack. Después de que se reciba el ack para un paquete saliente, el emisor desplaza la ventana de paquetes a lo largo del stream de bytes, y envía otro paquete. Este mecanismo de control se conoce como ventana deslizante.

Debido a la manera en que TCP funciona, no es realista usarlo como mecanismo para transportar la voz en una llamada VoIP. Con VoIP, la pérdida de paquetes es menos importante que la latencia. Actualmente, H.323 usa TCP, mientras que SIP y MGCP usa UDP (aunque SIP también soporta TCP como mecanismo de transporte).

Los campos de un paquete TCP son los siguientes:

- Puerto de origen y destino: identifica los puntos en los cuales el nivel superior, en el origen y en el destino, reciben los servicios TCP.
- Número de secuencia: especifica el número asignado al primer byte de datos del mensaje.
- Número de ack: contiene el número de secuencia del siguiente byte de datos que el enviador de datos espera recibir.
- Data offset: indica la cantidad de palabras de 32 bits en la cabecera TCP.
- Reserved: reservado para uso futuro.
- Flags: información de control.
- Windows: especifica el tamaño de la ventana de recepción del enviador.
- Checksum: indica si la cabecera a los datos se han dañado.
- Urgent pointer: apunta al primer byte de datos urgente en el paquete.
- Options: opciones de TCP variadas.
- Data: contiene información para el nivel superior.

8.3.2. UDP

UDP es un protocolo mucho más simple que TCP, y es útil en situaciones en las que los mecanismos de fiabilidad no son necesarios. UDP no es orientado a conexión y tiene una cabecera pequeña, lo que se traduce en una sobrecarga por el protocolo mínima.

La cabecera UDP tiene sólo cuatro campos: puerto de origen, puerto de destino, longitud del paquete y checksum, que es opcional.

UDP se usa en VoIP para portar el tráfico de voz, en lugar de TCP, cuyas características de control de flujo y retransmisión de paquetes, en el caso de audio en tiempo real, son innecesarias. Ya que UDP se usa para transporte del stream de audio, continuará haciéndolo independientemente de si está experimentándo un 5 ó un 50 por ciento de pérdida de paquetes.

Si se usase TCP en vez de UDP, la latencia que produciría los ack y las retransmisiones producirían una calidad de voz inaceptable. En VoIP, así como en otras aplicaciones en tiempo real, el control de latencia es más importante que asegurar la entrega de cada paquete.

TCP, no obstante, si es usado en el configuración de la llamada en la mayoría de protocolos VoIP.

8.3.3. RTP

Debido a la dependencia del tráfico de voz respecto al tiempo, UPD/IP es la elección lógica para transportar la voz. Sin embargo, se necesita más información que la que UDP ofrece para basar la comunicación en paquetes. Por tanto, para tráfico en tiempo real o sensible a retrasos, IETF (Internet Engineering Task Force) adoptó el protocolo RTP. VoIP usa RTP, el cual usa UDP. Por tanto, VoIP es transportada por la pila de protocolos RTP/UDP/IP.

RTP es el estandard para transmitir tráfico sensible a retrasos en redes basadas en paquetes. RTP da a los nodos receptores información que no está en los streams UDP/IP. Como se muestra en la Ilustración 8, dos bits importantes de información son el bit de secuencia y el timestamp. RTP usa la información de secuencia para determinar si los paquetes están llegando en orden, y usa la información de time-stamp para saber el tiempo de llegada entre paquetes (jitter).

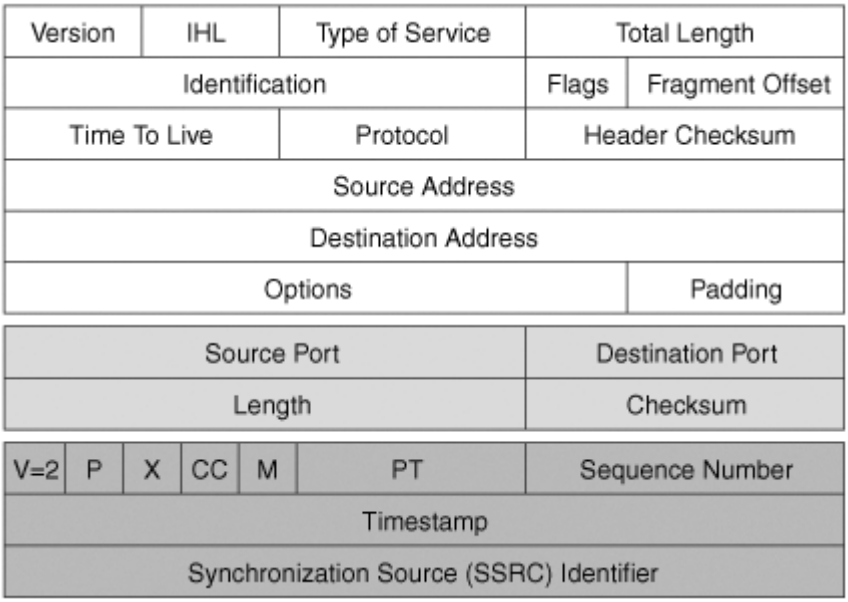


Ilustración 8: Cabecera del protocolo RTP

Junto a RTP se usa también RTCP (RTP Control Protocol) para enviar datos de control entre el emisor y receptor de una secuencia RTP. Los paquetes RTCP son enviados aproximadamente cada cinco segundos, y contienen datos que ayudan a verificar las condiciones de transmisión en el extremo remoto.

RTCP soporta también conferencias en tiempo real de grupos de cualquier tamaño en Internet. Esto incluye identificación del origen y soporte de gateways, como brigdes de audio y video, así como traductores de multicast a unicast. También ofrece realimentación de QoS desde los receptores al grupo multicast, así como soporte para la sincronización de diferentes streams de medios.

El uso de RTP es importante en tráfico en tiempo real, pero tiene sus inconvenientes. Las cabeceras de los protocolos IP/RTP/UPD son de 20, 8 y 12 bytes, lo que suman una cabecera de 40 bytes, que puede llegar a ser demasiado grande comparada con la carga útil. Se puede comprimir considerablemente usando RTP Header Compression (CRTP).

8.4. Latencia

Latencia en la VoIP es la cantidad de tiempo que tarda un sonido desde que sale de la boca del emisor hasta que llega al oído del receptor.

En las redes telefónicas de hoy hay tres tipos de retrasos: de propagación, de serialización y de gestión. El retraso de propagación está causado por la extensión que una señal debe recorrer por redes de fibra óptica o cobre. El retraso de gestión, también llamado de procesamiento, se refiere a varios tipos de retraso (paquetización, compresión y conmutación de paquetes) y está causado por dispositivos que reenvían la trama a través de la red.

El retraso de serialización es la cantidad de tiempo que se tarda en colocar un bit o un byte en un interface. Su influencia en la latencia es relativamente pequeña.

8.4.1. Retraso de propagación

La luz viaja a través del vacío a 300.000 kilómetros por segundo, y los electrones viajan a través de un cable de cobre o fibra aproximadamente a 200.000 kilómetros por segundo. Una red de fibra alrededor del mundo produce un retraso de unos 70 milisegundos. Aunque este retraso es casi imperceptible para el oído humano, en combinación con el retraso de gestión, puede causar una degradación en la voz considerable.

8.4.2. Retraso de gestión

Como se mencionó anteriormente, los dispositivos que envían tramas a través de la red pueden causar retrasos en la gestión. Estos retrasos en la gestión son un problema bastante mayor en las redes orientadas a paquetes que en las tradicionales redes telefónicas.

Por ejemplo, en algunos productos de Cisco para VoIP, el procesador de señales digitales (DSP) toma un sample de voz cada 10 milisegundos cuando usa G.729. Dos de estos samples, ambos con 10 ms de retraso, son incluidos en un paquete. Por tanto, el retraso del paquete es de 20 milisegundos. Además, G.729 tiene un retraso de 5 ms por look-ahead, por lo que se obtiene 25 ms de retraso inicial para la primera trama de voz.

Los vendedores pueden decidir cuantos samples quieren enviar en cada paquete, aunque en el caso de G.729, que toma samples cada 10 ms, cada incremento de samples eleva el retraso 10 ms.

8.4.3. Queuing Delay

Una red basada en paquetes experimenta retrasos por muchas razones. Uno de ellas es el tiempo necesario para mover un paquete a la cola de salida y a la cola de espera.

Cuando los paquetes están en una cola debido a la congestión de un interfaz de salida, el resultado es un retraso por encolamiento (queuing delay). Esta situación ocurre cuando se envían más paquetes de los que el interfaz puede gestionar en un determinado intervalo.

Se debe mantener este retraso por debajo de 10 ms siempre que se pueda, usando el método de encolado que sea óptimo para cada red.

La organización ITU-T, en su recomendación G.114, especifica que para tener una buena calidad de voz no debe haber un retraso de más de 150 ms de extremo a extremo.

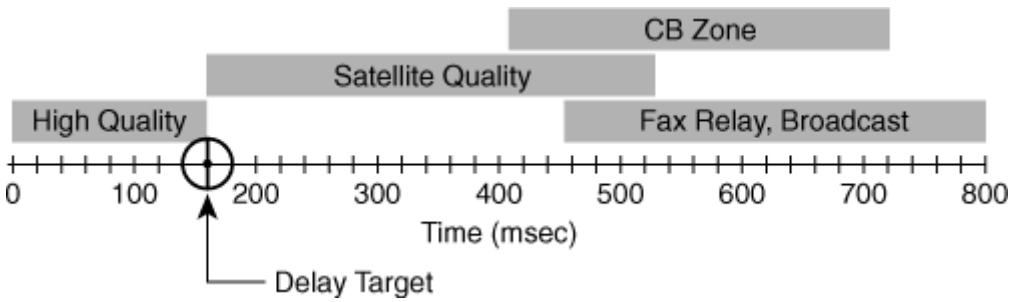


Ilustración 9: Retraso End-to-End

Como se muestra en la Ilustración 9, aunque algunos retrasos son grandes, se aceptan porque no hay otra alternativa. Por ejemplo, en transmisión por satélite se tarda aproximadamente 250 ms en alcanzar un satélite, y otros 250 ms en bajar a la Tierra. Aunque este tiempo, 500 ms, excede la recomendación de la ITU-T, en la práctica estos enlaces se utilizan bastante, por lo que en realidad se suele definir la calidad de voz como lo que los usuarios están dispuestos a aceptar.

En una red congestionada o no bien gestionada, el retraso por queuing puede añadir hasta dos segundos de retraso. Este valor es inaceptable en la mayoría de las redes de VoIP.

8.5. Jitter

El Jitter es la variación del tiempo de llegada entre paquetes. Este problema existe solamente en redes basadas en paquetes. El emisor de paquetes puede transmitir de manera fiable paquetes de voz a intervalos regulares (por ejemplo, una trama cada 20 ms). Estos paquetes de voz pueden ser retrasados a lo largo de su recorrido por la red de paquetes, y no llegar al receptor en el mismo intervalo regular (ver Ilustración 10). La diferencia entre el tiempo en el que se esperaba el paquete y cuando se recibió realmente es el jitter.

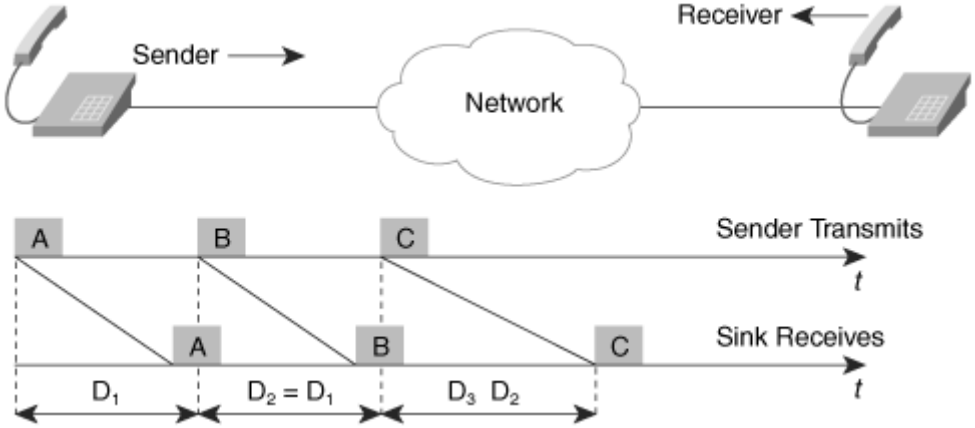


Ilustración 10: Variación del tiempo de llegada de un paquete (Jitter)

En la Ilustración 10, se observa que la cantidad de tiempo que tardan los paquetes A y B en el emisor y el receptor son iguales (D1=D2). Sin embargo, el paquete C encontró retraso en la red, por lo que su recepción fue posterior a la esperada. Esta es la razón por lo que es necesario un buffer de jitter, que corrija las variaciones de retrasos entre paquetes. Se recomienda crear un ratio de los paquetes que llegan tarde para poder ajustar adecuadamente el buffer de jitter.

Como los buffer de jitter no pueden ser infinitos, a veces es mejor descartar paquetes o tener buffers de tamaño fijo. Si las redes de datos están bien gestionadas y diseñadas, el jitter no debe ser un problema grave, por lo que los buffer de jitter no contribuirán significativamente al retraso total.

8.6. Modulación por impulsos codificados (PCM)

Aunque la comunicación analógica es ideal para los humanos, la transmisión analógica no es

robusta ni eficiente para recuperarse de ruidos de línea. En los orígenes de las redes telefónicas, cuando las transmisiones analógicas pasaban por amplificadores para reforzar la señal, no sólo se incrementaba la voz sino también el ruido. Este podía llegar a provocar que la comunicación fuese imposible. Es mucho más sencillo separar el ruido de samples digitales, compuestos de 1 y 0. Por tanto, cuando se regenera una señal analógica a partir de samples digitales, todavía se mantiene un sonido claro.

PCM convierte sonido analógico en formato digital muestreando la señal analógica 8.000 veces por segundo, y convirtiendo cada muestra (o sample) en un número. El teorema de Nyquist establece que se debe muestrear una señal al doble de la frecuencia más alta que interesa, para después poder reconstruir la señal analógica fielmente. Debido a que la voz humana está casi toda por debajo de 4kHz, se necesita una frecuencia de muestreo de 8kHz (125 microsegundos entre samples).

La cantidad de samples que se envían por trama dependerá del codec que se elija, así como del compromiso entre ancho de banda e impacto de la pérdida de paquetes. Mientras mayor sea el número de samples, mejor utilización de ancho de banda, ya que se envía más información en el payload del paquete UDP/RTP, y por tanto la sobrecarga de la red por tráfico de cabeceras es menor. No obstante, el impacto por la pérdida de uno de esto paquetes es mayor, ya que van más muestras en él.

En la tabla 2 se muestra el número de samples por trama para diferentes codecs:

Codec	Samples por Frame (Por defecto)	Samples por Frame (Máximo)
PCMU/PCMA	2	10
G.723	1	32
G.726-32	2	20
G.729	2	64
G.728	4	64

Tabla 2: Samples por frame para distintos codecs

8.7. Compresión de la voz

Para comprimir la voz, se usan dos variaciones básicas de PCM a 64 Kbps, μ-law and a-law. Ambos métodos son similares en que ambos usan compresión logarítmica para conseguir de 12 a 13 bits de calidad en 8 bits, pero se diferencian en detalles menores. Por razones históricas, se usa μ-law en Estados Unidos y a-law en Europa. Cuando se realizan llamadas de larga distancia, puede ser necesario hacer una conversión entre ambos formatos.

Otro método de compresión usado frecuentemente es Adaptive Differential Pulse Code Modulation (ADPCM). Una implantación de ADPCM es ITU-T G.726, que codifica usando muestras de 4 bits, dando una ratio de transmisión de 32 Kbps. A diferencia de PCM, los 4 bits no codifican directamente la amplitud de la voz, sino las diferencias en la amplitud, así como el ratio de cambio de dicha amplitud.

PCM y ADPCM son ejemplos de codecs de ondas, que usan técnicas de compresión que explotan características redundantes de las propias ondas. Se han desarrollado nuevas

técnicas de compresión en los últimos 10 ó 15 años que explotan el conocimiento de la generación de la voz. Estas técnicas pue incluir variaciones como Linear Predictive Coding (LPC), Code Excited Linear Prediction Compression (CELP), y Multipulse-Multilevel Quantization (MP-MLQ).

8.7.1. Voice Coding Standards

ITU-T ha estandarizado CELP, MP-MLQ PCM y ADPCM en su serie G de recomendaciones. Las más populares para la telefonía por paquetes son:

- G.711: describe la codificación de voz de 64 Kbps PCM descrita anteriormente. La voz codificada de esta manera está ya en el formato correcto para entregarla digitalmente a la red telefónica o a una Private Branch eXchanges (PBX).
- G.726: describe la codificación ADPCM a 40, 32, 24 y 16 Kbps.
- G.728: describe la variación de compresión CELP a 16 Kbps para retrasos pequeños.
- G.729: describe la compresión CELP que permite streams de voz de 8 Kbps.
- G.723.1: describe una técnica de compresión que se puede usar para comprimir voz o sonido multimedia a un bajo nivel de rate, como parte de la familia de standares H.324. Se definen dos rates, a 5,3 Kbps, asociado a CELP, y a 6,3 Kbps, asociado a MP-MLQ y que proporciona mayor calidad.
- ILBC (Internet Low Bitrate Codec): un codec de voz libre capaz de ofrecer una comunicación de calidad sobre IP. Está diseñado para poco ancho de banda, teniendo un rate de payload de 13,33 kbps con una separación entre tramas de 30 ms, y de 15,2 kbps con una separación de 20 ms. Es usado por muchas aplicaciones en PC, como Skype, Google Talk, Yahoo Messenger y MSN Messenger.

8.7.2. Mean Opinion Score

Se puede testear la calidad de los anteriores codecs de dos formas: objetiva y subjetivamente. Los test subjetivos son realizados por personas, mientras que los computadores hacen los test objetivos. Normalmente los codecs son desarrollados y afinados basándose en medidas subjetivas de la calidad de voz.

Un benchmark subjetivo muy común para cuantificar el rendimiento del codec es el Mean Opinion Score (MOS). Los test MOS se le dan a un grupo de oyentes. Es importante que se elija un rango de oyentes y de samples lo suficientemente grande para hacer el test. Los oyentes califican la calidad de cada sample puntuando de 1 (malo) a 5 (excelente). Al final, se calcula la media de todas las las puntuaciones.

En la tabla 3 se muestran las puntuaciones de varios codecs de ITU-T:

Método de compresión	Bit Rate (Kbps)	Sample Size (ms)	MOS Score
G.711 PCM	64	0.125	4.1
G.726 ADPCM	32	0.125	3.85
G.728 Low Delay Code Excited Linear Predictive (LD-CELP)	15	0.625	3.61
G.729 Conjugate Structure Algebraic Code Excited Linear Predictive (CS-ACELP)	8	10	3.92
G.729a CS-ACELP	8	10	3.7
G.723.1 MP-MLQ	6.3	30	3.9
G.723.1 ACELP	5.3	30	3.65
iLBC Freeware	15.2	20	3.9

Tabla 3: Puntuaciones en el test MOS

8.8. Comparativa establecimiento de llamada PSTN/VoIP

Esta sección muestra el proceso necesario para establecer una llamada en la red PSTN, así como en usando VoIP. Para simplificar, se supone que se está llamando por teléfono a un vecino.

En caso de la llamada PSTN, el proceso sería así:

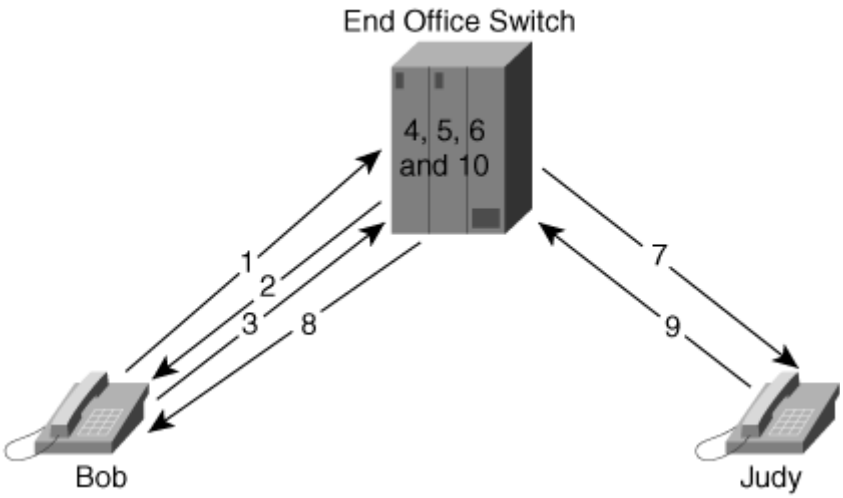


Ilustración 11: Llamada PSTN

En este ejemplo, Bob llama a su Vecina Judy. Ambos son abonados y están en la misma centralita. Son necesarios los siguientes pasos:

1. Bob descuelga su teléfono.
2. La centralita local da a Bob tono de marcación.
3. Bob marca el número de Judy.
4. La centralita analiza el número que ha marcado Bob para determinar si el destino de la llamada es local o remoto.
5. El switch de la centralita identifica el número de Judy.
6. El switch de la centralita envía por la línea de Judy la señal de llamada.
7. A Bob se le envía un señal de retorno por la que puede oir los tonos de marcación que se están enviando a Judy.
8. Judy descuelga su teléfono.
9. El switch de la centralita establece el circuito entre Bob y Judy.

Si el switch de la centralita de Bob no sirviese también a Judy, el switch de la centralita de Bob enrutaría la llamada a otro switch que pueda determinar como contactar con Judy.

La Ilustración 12 muestra el proceso para completar una llamada desde una aplicación de PC a otra:

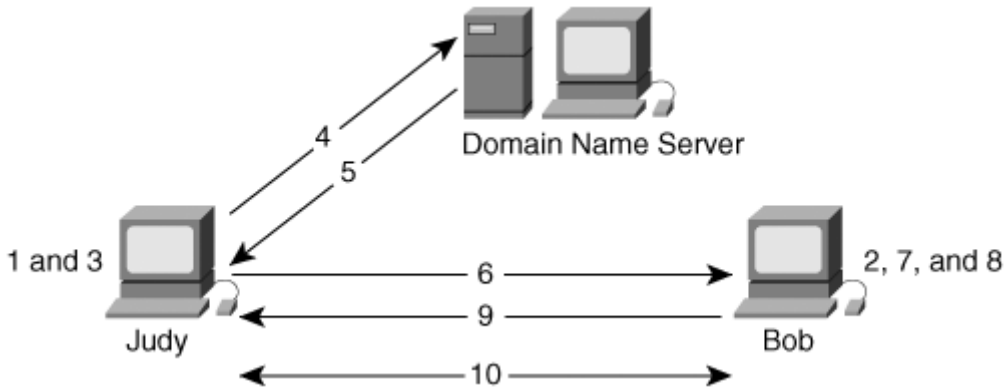


Ilustración 12: Llamada con un teléfono IP

Bob y Judy necesitan estar ambos conectados a la misma red IP, o estar ambos conectados a Internet. El proceso sería el siguiente:

1. Judy inicia su aplicación de telefonía de Intener (I-phone), que sea compatible con H.323.
2. Bob ya tiene su aplicación I-phone lanzada.
3. Judy sabe que el nombre de Bob en Intenet es bob@nextdoorneighbor.com, así que pone está dirección en el campo "a quien llamar" de su aplicación I-phone, y pulsa "Llamar".
4. La aplicación I-phone convierte la dirección bob@nextdoorneighbor.com a un nombre de host, y va a un servidor DNS para averiguar su IP.
5. El servidor DNS le devuelve a Judy la IP de Bob.

6. La aplicación I-phone de Judy coge la dirección IP de Bob y el envía un mensaje usando H.225.
7. El mensaje H.225 señaliza en el PC de Bob la señal de llamada.
8. Bob acepta el botón de "Aceptar", que indica a la aplicación I-phone que envíe a Judy un mensaje H.225 de conexión.
9. La aplicación I-phone de Judy comienza la negociación H.245 con la de Bob.
10. La negociación H.245 termina y los canales lógicos se abren. Bob y Judy pueden hablar entre sí a través de una red basada en paquetes.

Como IP es un protocolo extendido por todo el mundo, cuando una llamada se paquetiza puede ser destinada de igual manera a tu vecino o a cualquier país del mundo.

8.9. Servicios de facturación y mediación

Los servicios de facturación y mediación son muy importantes en VoIP, ya que ayudan a un proveedor de servicio o a un vendedor empresarial a gestionar aspectos financieros importantes, como el Retorno de la Inversión (ROI), cuando se migra desde redes basadas en multiplexión del tiempo (TDM) a VoIP. La red PSTN ofrece una estructura de facturación simple, ya que el origen y el destino de la llamada están unidos a una localización física, además de tener una facturación por minutos de voz de la llamada, o por cantidad de tráfico en caso de tráfico de datos.

VoIP cambia este paradigma, permitiendo que los extremos de la comunicación se muevan. El tráfico de voz y datos son ambos paquetes que son transportados de una localización a otra a través de una red IP. Esto abre algunas cuestiones y requiere protocolos adecuados para definir a quien cobrar, cuando y qué cobrar.

Los servicios de mediación y facturación son, por tanto, piezas importantes de cualquier servicio VoIP. Con tantos protocolos VoIP y tantos proveedores lanzando sus propio servicios, hay que definir procedimientos de cobro estandarizados antes de poner en marcha el servicio.

Alguna de las prestaciones por las que se puede definir facturación específica son:

- Facturación por duración (por ejemplo, llamadas de voz).
- Facturación por bytes de datos (por ejemplo, para modems).
- Facturación por página (por ejemplo, para fax).
- Facturación por tarifa plana.
- Roaming: integración con los servicios de facturación de los partners.
- Conferencias.
- Servidor de Voice-mail.

8.9.1. Autenticación, autorización y accounting (AAA)

AAA es una parte fundamental de los servicios de pago en el mundo IP. Los clientes son entidades que tienen el control de la llamada, siendo los servidores donde se procesa la información relativa a la facturación. Normalmente, existen servidores dedicados al control de la facturación, para los que los servidores de VoIP son clientes que usan sus servicios. Los servicios ofrecidos son:

1. Autenticación: proporciona un medio para identificar a un cliente que solicita acceso a algún sistema. Se realiza mediante el intercambio de claves o certificados entre el cliente y el servidor.
2. Autorización: posterior a la autenticación, determina si el cliente está autorizado a realizar la operación solicitada.
3. Accounting: es el proceso de medir el consumo de recursos, permitiendo la monitorización y el reporting de eventos, ya sea para la facturación o el análisis. VoIP permite nuevos modelos de accounting debido a sus características como la movilidad y la facilidad para transportar el tráfico de voz sobre datos.

Una solución frecuente usada por los proveedores de VoIP es el protocolo RADIUS (Remote Authentication Dial-In User Service), que está diseñado para dotar de seguridad, así como de recolección de estadísticas, a un entorno remoto de computación, como puede ser la red distribuida de VoIP. Permite realizar el accounting de manera centralizada, lo cual es siempre más fiable y fácil de manejar que un montón de datos repartidos por múltiples dispositivos.

RADIUS permite separar en distintas máquinas las funciones de servidor de autenticación y de recolección de datos, y los administradores de red pueden configurar los clientes RADIUS para que usen ambos servicios o sólo uno de ellos. Un sólo servidor RADIUS puede dar servicio a cientos de clientes RADIUS, y se puede añadir tolerancia a fallos y redundancia configurando el cliente para que use servidores RADIUS alternativos.

El servidor de accountig de RADIUS actúa como almacén CDR (Call Data Record, o Registro de Datos de Llamadas). Puede ser ampliado por extensiones de cada vendedor para añadir la recolección de datos adicionales para VoIP, wireless, prepago, o cualquier otro servicio sobre las llamadas.

Los gateways VoIP, los softswitches y los servidores intermedios tienen la responsabilidad de de proporcionar a servidores de accounting de RADIUS la información necesaria, tales como las IP de origen y destino, los puertos, la duración de la llamada, etc

8.10.Seguridad

En este capítulo se da una visión global de los requisitos de seguridad típicos de un servicio de VoIP.

8.10.1. Requisitos de seguridad

Antes de entrar en detalle en las técnicas disponibles para hacer segura una red VoIP, es necesario entender los principales problemas y requisitos de seguridad que se deben cumplir:

- Integridad: se debe garantizar que el receptor recibe los paquetes tal cual se los enviaron, sin modificación. Para la parte de la señalización de VoIP, además, no debe haber pérdida de paquetes.
- Privacidad: un tercero no debe poder leer datos que no vayan dirigidos a él.
- Autenticidad: tanto el emisor con el receptor de una llamada VoIP deben estar seguros de que se están comunicando con quien dice ser.
- Disponibilidad: el servicio debe estar disponible sin interrupciones para todos sus usuarios.

8.10.2. Tecnologías de seguridad

Dados los requisitos anteriores, varias de las tecnologías que podemos usar para lograr integridad, privacidad y autenticidad son:

8.10.2.1. Seguridad por clave compartida (Shared-Key)

En esta alternativa, ambas partes de la comunicación comparten una clave (shared-key), que es desconocida por otras partes.

Como inconvenientes, el administrador debe provisionar la clave compartida a todos los dispositivos, lo que en un entorno grande puede ser prohibitivo. Además, si la clave fuese comprometida, se debería reasignar a todos los dispositivos una nueva.

8.10.2.2. Criptografía de clave pública

Para solucionar los problemas de la clave compartida se puede usar criptografía de clave pública. Los conceptos clave de este tipo de criptografía son las claves asimétricas y las firmas digitales.

8.10.2.2.1. Claves asimétricas

Un par de claves asimétricas son dos números, uno privado que no se le da a nadie, y otro público, con una relación matemática entre ambos. Estos números cumplen las propiedades:

- Unos datos encriptados con una clave privada sólo pueden ser desencriptados con su clave pública.
- Unos datos encriptados con una pública sólo pueden ser desencriptados con su clave privada.
- Hay una relación de uno-a-uno entre claves.

De esta manera, para conseguir autenticación, en emisor puede usar su propia clave privada para encriptar un mensaje, usando el receptor la clave pública de emisor para desencriptarlo. Para una comunicación segura, el emisor encripta con la clave pública del receptor, pudiendo ser desencriptado el mensaje sólo con la clave privada del receptor.

Como los procesos de encriptado/desencriptado con claves asimétricas consumen mucha CPU, se utiliza sólo para negociar una clave compartida para esa sesión, usándose esta clave

para el resto de la sesión.

8.10.2.2.2. Digital Signature

Una firma digital es un atributo del contenido del mensaje y el firmante de dicho mensaje. Sirve para un propósito similar que una firma en el mundo real, autenticar un mensaje o un fragmento de información.

El emisor calcula una función hash sobre el contenido del mensaje, el cual es firmado con la clave privada del emisor. Este fragmento es la firma digital, y es añadida al mensaje.

El receptor desencripta el mensaje con la clave pública del emisor, y como ha recibido también el mensaje original, vuelve a calcular la función hash, comparando el resultado con lo recibido.

Las firmas digitales proporcionan autenticación e integridad, pero no privacidad, ya que la firma se añade al mensaje, que se transmite en abierto.

8.10.2.3. Certificados y Autoridades de Certificación

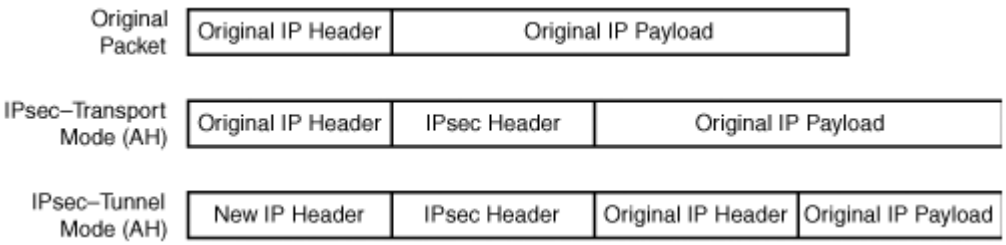
Llegados a este punto, queda pendiente la cuestión de como propagar la clave pública de un dispositivo a todos los posibles receptores. La solución viene por el uso de certificados. La generación de un certificado comienza enviando la clave pública del dispositivo a una Autoridad de Certificación (CA), la cual certifica la identidad del emisor y emite un certificado que garantiza la relación entre el solicitante y su clave pública.

Cada dispositivo tendrá su propio certificado, y será usado en todas sus comunicaciones, enviándolo al inicio de la sesión. El otro extremo verificará que la CA realmente firmó el certificado, para lo que necesitará tener previamente guardada la clave pública de la CA, y almacenará en caso de validarse correctamente la clave pública transmitida con el certificado para usarla en la comunicación con ese dispositivo.

8.10.2.4. Protocolos de seguridad basados en clave pública

Algunos de los protocolos de propósito general que usan clave pública son:

- TLS: definido en la RFC 2246, es una evolución de SSL. TLS se ubica por encima de un protocolo de transporte fiable, como TCP, y es independiente del nivel de aplicación que lo use. En VoIP se suele usar para securizar la señalización.



- IPsec: opera en el nivel IP y proporciona seguridad a los datagramas IP. Usan dos protocolos, Authenticacion Header (AH), que proporciona autenticación e integridad, y Encapsulation Security Payload (ESP), que además proporciona privacidad del mensaje.
- Secure Real-time Transport Protocol (SRTP): definido en la RFC 3711, proporciona

integridad, autenticación y privacidad al tráfico RTP y al de control, RTCP. SRTP no especifica como se produce el intercambio de claves entre el emisor y el receptor, por lo que debe usar TLS, IPsec u otro protocolo con idéntico fin antes de empezar a usar SRTP.

8.10.2.5. Protegiendo dispositivos de Voz

Para la disponibilidad del servicio IP, es necesario que los dispositivos involucrados estén protegidos contra ataques. Algunas de las medidas básicas que se pueden tomar son:

- Deshabilitar servicios y puertos sin uso: normalmente todo dispositivo tienen servicios y puertos abiertos, haciéndolos vulnerables a ataques de hackers. Lo recomendable es desactivar estos servicios y puertos de los dispositivos de VoIP y de los dispositivos de infraestructura (switches, routers, etc). Algunos de los servicios candidatos a ser desactivados son TFTP, Telnet y SNMP.
- Host-based Intrusion Protection Systems (HIPS): se puede usar HIPS para securizar dispositivos críticos en la infraestructura. Este sistema monitoriza el uso de CPU, los intentos de login, etc., y si encuentra valores anómalos adopta las medidas pertinentes, que pueden ser desde limitar tráfico desde alguna IP a terminar aplicaciones consideradas sospechosas.

8.10.2.6. Protegiendo la infraestructura de red IP

Como el servicio de VoIP funciona en una red, no es suficiente con proteger los dispositivos de voz, sino también la propia red que transporte el tráfico. A continuación se presenta una serie de medidas básicas que siguen este fin:

- Segmentación: el tráfico VoIP se debe separar de otro tipo de tráfico, al ser un servicio crítico. Esto se puede conseguir con VLANs, en el nivel 2, definiendo una VLAN para las llamadas de voz, o en el nivel 3, usando diferente direccionamiento IP para VoIP.
- Políticas de tráfico: un sólo dispositivo VoIP puede consumir todo el ancho de banda destinado al tráfico VoIP, afectando por tanto a otros dispositivos. Para prevenir esto, se necesita implementar políticas de tráfico.

  La mayoría de dispositivos VoIP generan un tráfico máximo limitado. Por ejemplo, teléfonos que usen el codec g.711 generan tráfico de no más de 64 kbps para cada dirección de la voz, por lo que se puede usar esta información para limitar el tráfico proveniente de cada dispositivo, de manera que se prevenga que un dispositivo averiado colapse la red.
- Autenticación de dispositivo 802.1x: es necesario bloquear accesos no autorizados a la red, ya que la denegación a los dispositivos a nivel 2 es la primera línea de defensa de una red. 802.1x y Extensible Authentication Protocol (EAP) son estándares para controlar el acceso por puerto, tanto para acceso cableado como wireless. Mediante estos protocolo se obliga a que un dispositivo se autentifique antes de acceder a un switch. EAP permite mecanismos de autentificación como RADIUS. Como es lógico, los dispositivos deben ser compatibles con este mecanismo de autenticación.

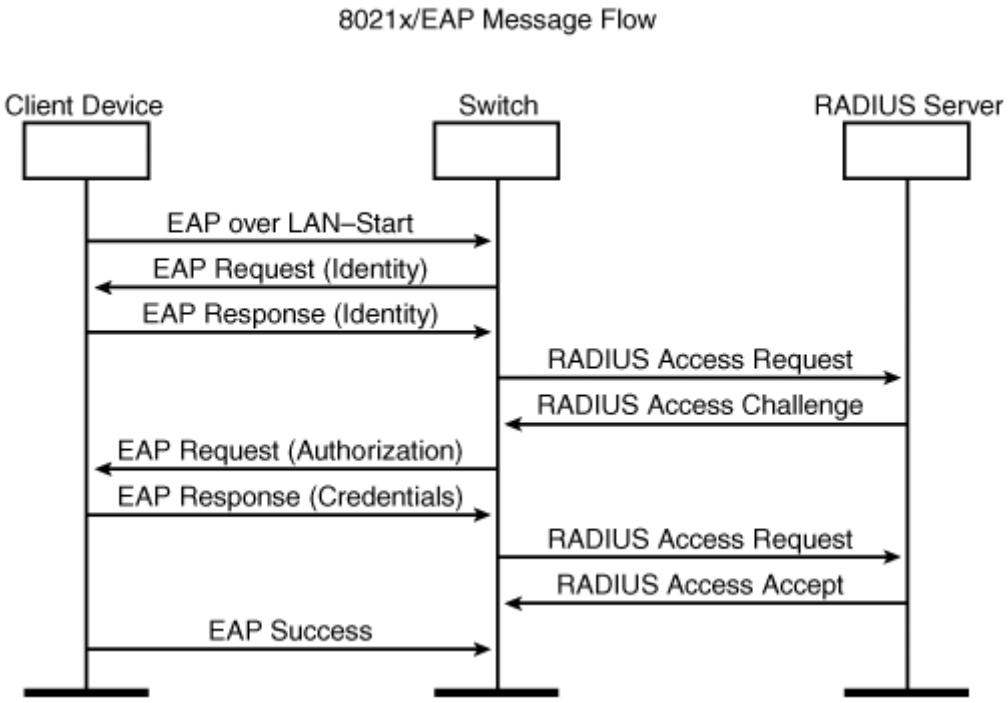


Ilustración 13: Proceso de autenticación 802.1x con RADIUS

Es necesario llegar a un compromiso entre la complejidad de la red y el riesgo asumido cuando se implementa una nueva tecnología. Por ejemplo, las técnicas de clave pública implican una sobrecarga de trabajo inicial al desplegar la infraestructura, por la emisión de certificados, etc. Sin embargo, una vez montada, una infraestructura PKI necesita un mantenimiento día a día mínimo.

8.11.Protocolos de señalización IP

8.11.1. H.323

H.323 es una especificación de ITU-T para transmitir video, audio y datos a través de una red IP. H.323 se encarga de la señalización y el control de llamadas y transporte de datos multimedia, así como controlar el ancho de banda punto a punto en multiconferencias. Actualmente, la última versión de H.323 es la versión 7.

8.11.1.1.Componentes de H.323

La siguiente figura ilustra los elementos de un sistema H.323.

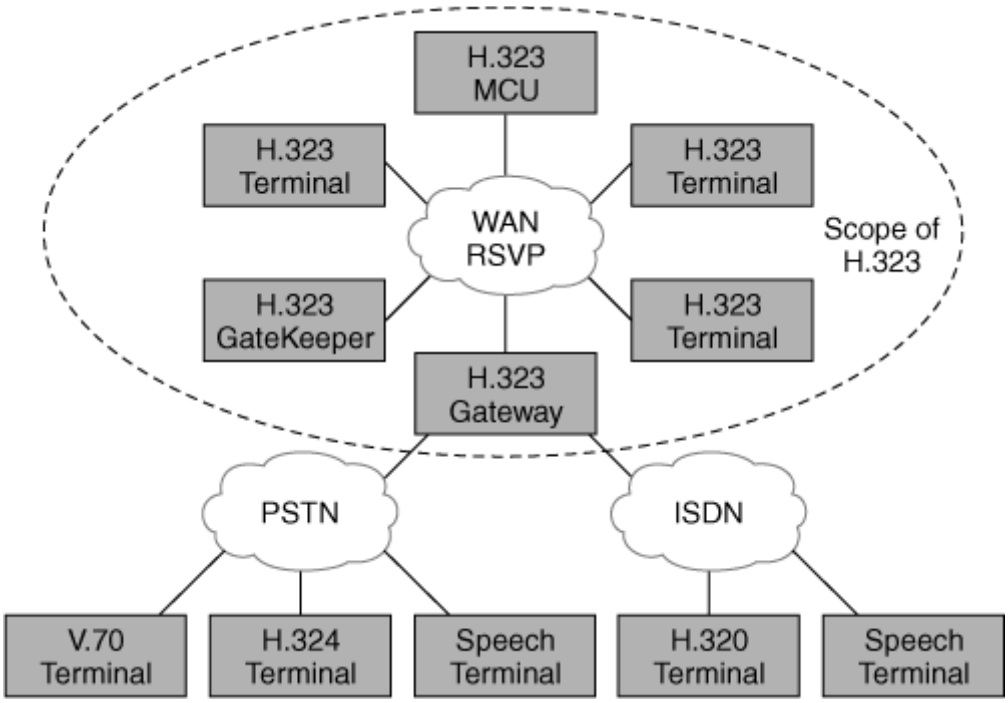


Ilustración 14: Elementos de un sistema H.323

Los Gateways conectan terminales H.323 a redes PSTN o RDSI. Los Gatekeepers realizan el control de admisión y traducción de direcciones para los terminales y los gateways. Los MCU son dispositivos que permiten a dos o más terminales o gateways hacer conferencia con audio o video.

8.11.1.2.Terminal

En la siguiente figura se muestra la definición de H.323 de un terminal:

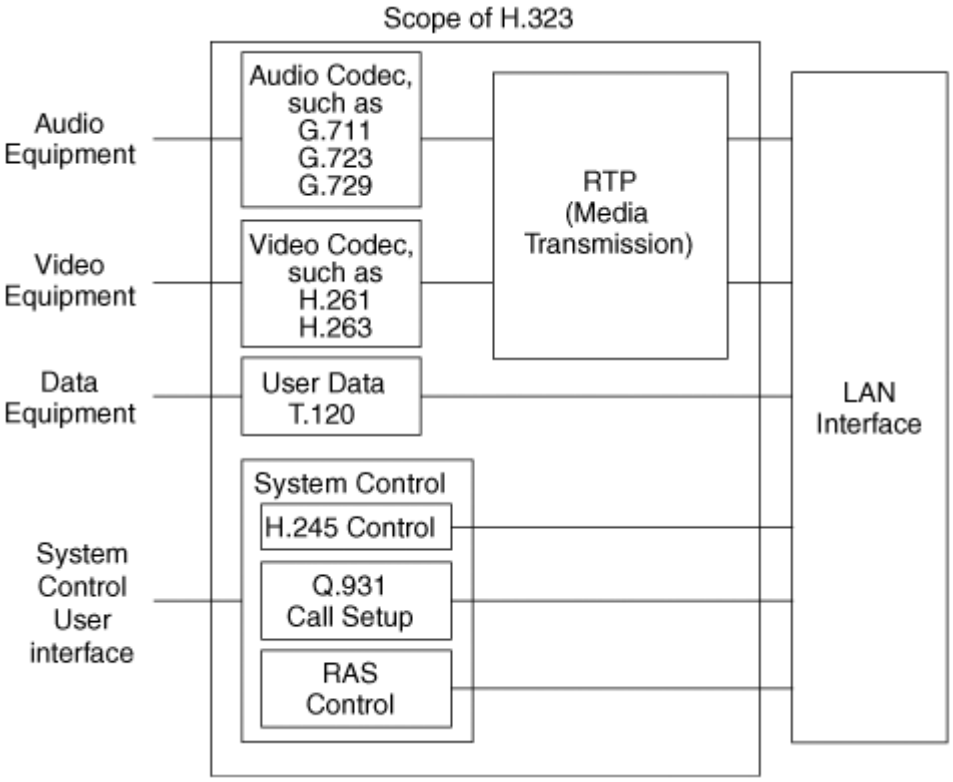


Ilustración 15: Terminal según H.323

Las siguientes funciones y capacidades están dentro del ámbito de un terminal H.323:

- System Control Unit: mediante H.225 y H.245 proporciona control de llamada, mensajería y señalización de comandos para el funcionamiento del terminal.
- Media Transmission: formatea el audio, video, datos, mensajes y ordenes de control al interface de red. Tambien se encarga de recibir estos tipos de tráfico.
- Codecs de audio: ofrece el uso de G.711, transmitiendo en a-law o μ-law. Opcionalmente, los codecs G.722, G.723.1, G.728 y G.729 pueden ser usados.
- Codec de video: aunque es opcional, si se proporciona debe ser capaz de codificar y decodificar video según los estándares H.261 y H.263.

8.11.1.3.Gateway

Un gateway H.323 refleja las características de un terminal SCN (Switched Circuit Network) y un terminal H.323. Traduce los formatos de audio, video y datos, así como los protocolos de comunicación. Los gateways no son necesarios a no ser que se requiera interconexión de redes SCN y VoIP.

La siguiente ilustración muestra como encaja un gateway:

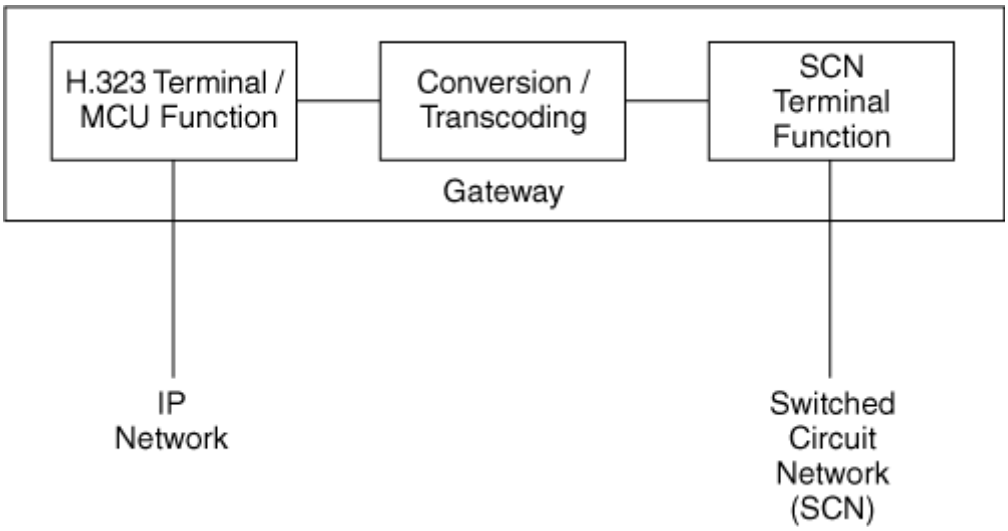


Ilustración 16: ubicación de Gateway

8.11.1.4. Gatekeeper

Como función opcional, el gatekeeper provee control de servicios antes y durante la llamada a los terminales H.323. Si está presente en un sistema H.323, debe hacer lo siguiente:

- Traducción de direcciones: traduce alias H.323 o direcciones E.164 a direcciones IP de dispositivos.
- Control de admisión: provee autorización al sistema usando mensajes de petición/confirmación/denegación de admisión (ARQ/ACF/ARJ).
- Control de ancho de banda: gestiona los requisitos de ancho de banda mediante mensajes de petición/admisión/denegación de ancho de banda.

8.11.1.5. Multipoint controller (MCU)

Un multipoint controller (MC) permite conferencias entre tres o más terminales en una conferencia multipunto. Recibe audio, video o data streams y lo distribuye por los terminales participantes en la conferencia, transmitiendo las capacidades de cada terminal, y revisándolas durante la conferencia.

8.11.2. Suite de protocolos H.323

La familia de protocolos que componen H.323 soportan la admisión de llamadas, la configuración, el estado, streams multimedia y mensajes en un sistema H.323. Estos protocolos son soportados en redes de datos fiables y no fiables.

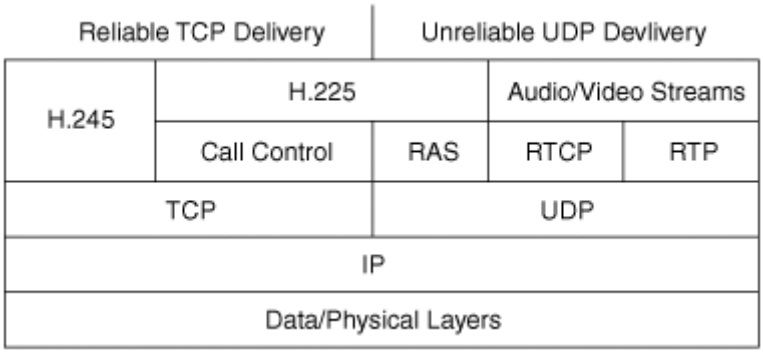


Ilustración 17: protocolos H.323 por niveles

Aunque la mayoría de las implementaciones actuales de H.323 utilizan TCP como protocolo de transporte para señalización, H.323 versión 2 habilita su transporte por UDP.

La suite de protocolos H.323 se divide en tres areas de control:

- Señalización de registro, admisión y estado: proporciona control pre-llamada en redes basadas en gatekeepers H.323.
- Señalización de control de llamada: usada para conectar, mantener y desconectar llamadas entre dos terminales.
- Transporte y control de medios: proporciona un canal fiable H.245 que transporta mensajes de control multimedia, usando el protolo no fiable UDP.

La figura siguiente muestra el proceso de establecimiento de llamada entre dos terminales usando H.323. Se asume que ambos terminales se han registrado en el mismo gatekeeper:

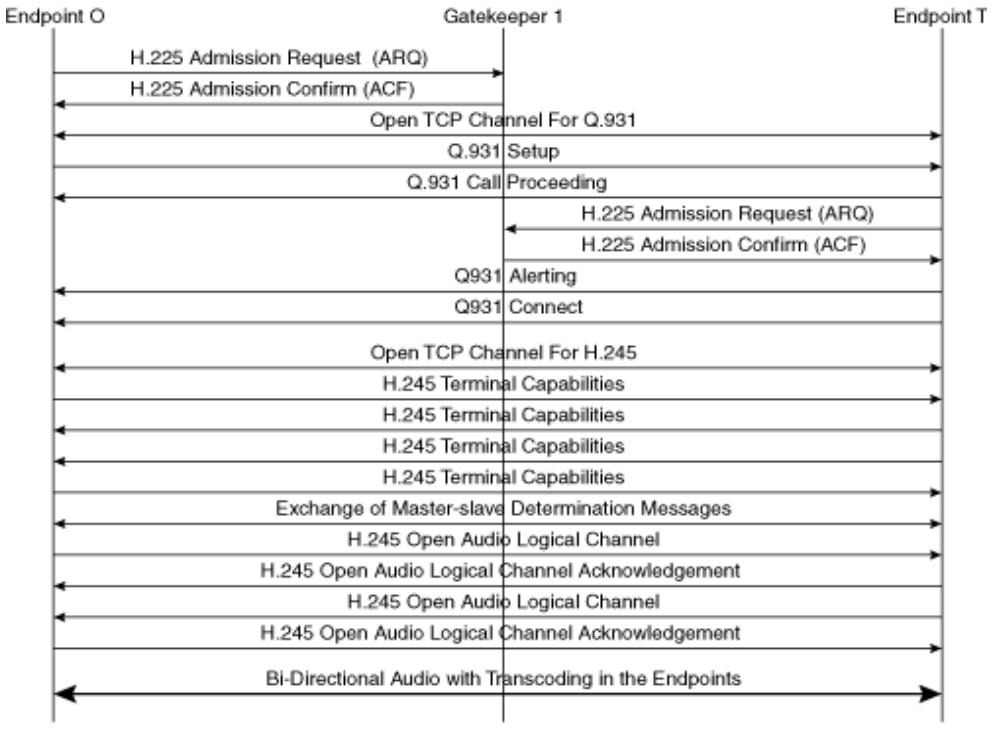


Ilustración 18: establecimiento de llamada H.323

Aunque el estandar H.323 cada vez es más completo según avanzan las revisiones, se han detectado algunos problemas, como tiempos de configuración de llamada excesivos, problemas con el protocolo para conseguir conferencias con todas las características, demasiadas funciones requeridas en cada gatekeeper, y problemas de escalabilidad en los gatekeepers.

Para configuraciones de terminales inteligentes, el protocolo SIP resuelve alguno de los problemas encontrados en H.323, implantándose como alternativa en muchas redes. De todas formas, H.323 todavía domina una parte sustancial de los despliegues de VoIP en el mercado de proveedores de voz.

8.11.3. Protocolo SIP

Session Initiation Protocol (SIP) es un protocolo que controla el inicio, modificación y terminación de sesiones multimedia interactivas. Las sesiones pueden ser de audio o video entre dos o más partes, sesiones de chat o de juegos. Se han definido también extensiones SIP para mensajería instantánea y notificación de eventos. SIP es un protocolo basado en texto similar a HTTP o SMTP.

SIP es un protocolo punto a punto, lo que significa que capacidades de red como enrutado de llamadas y gestión de sesiones están distribuidas por todos los nodos, incluidos los terminales y los servidores de red. SIP es extensible, por lo que se puede ir incrementando su funcionalidad conforme vayan surgiendo nuevas prestaciones en la telefonía IP.

8.11.3.1. Functionalidad de SIP

SIP proporciona las siguientes capacidades para permitir sesiones multimedia:

- Localización de usuarios: SIP provee la funcionalidad de descubrir la localización del usuario final para establecer una sesión con él o entregarle una petición SIP. La movilidad del usuario está soportada en SIP de manera inherente.
- Capacidades de los usuarios: SIP permite determinar las capacidades multimedia de los dispositivos involucrados en una sesión.
- Disponibilidad de usuarios: permite fijar si el usuario desea tomar parte en una comunicación.
- Configuración de sesión: SIP habilita el establecimiento de parámetros de sesión para todas las partes de la comunicación.
- Manejo de sesión: SIP permite la modificación, transferencia y terminación de una sesión activa.

8.11.3.2. Elementos de red de SIP

Una red SIP consta de los siguientes dispositivos:

- User Agent (UA): es una función lógica en la red SIP que inicia o responde a una transacción SIP. Una UA puede actuar como cliente o como servidor en una transacción SIP, y mantiene un estado de la sesión.

- User Agent Client (UAC): es una función lógica que inicia peticiones SIP y acepta respuestas SIP. Por ejemplo, SIP phones iniciando una llamada.
- User agent server (UAS): es una función lógica que acepta peticiones SIP y envía respuestas SIP. Por ejemplo, un teléfono SIP aceptando una petición INVITE.
- Proxy: es un intermediario entre la red SIP del UAS y otro proxy de la red SIP del UAC, y ejerce principalmente funciones de enrutado en la red SIP.
- Redirect Server: es un UAS que genera respuestas SIP a las peticiones que recibe, diciéndole al UAC que contacte con otro URI (Uniform Resuource Identifier).
- Registrar server: es una UAS que acepta peticiones SIP REGISTER y actualiza la información sobre el mensaje en una base de datos de localización.

8.11.3.3. Interacción con otros protocolos IETF

SIP por sí mismo no proporciona todas las capacidades requeridas para configurar una sesión multimedia interactiva. En lugar de eso, SIP es un protocolo que forma parte de un framework de protocolos estándares que constituyen una arquitectura multimedia.

Los agentes y aplicaciones SIP necesitan otros protocolos para lo siguiente:

- Describir las características de una sesión, como si es de audio o vídeo, codecs a usar, etc.
- Manejar media: estos protocolos controlan y transmiten paquetes de audio y video para una sesión.
- Otras funciones, como AAA (autenticación, autorización y accounting), TLS, DNS para resulución host a IP, etc.

Normalmente, las sesiones SIP usan los siguientes protocolos IETF:

- DNS: una sesión SIP puede necesitar el uso de DNS para resolver nombres de host a IP.
- Session Description Protocol (SDP): se usa en el cuerpo de un mensaje SIP para describir los parámetros de una sesión multimedia.
- Real-time Transport Protocol: transporta datos en tiempo real, como audio o vídeo, a los terminales involucrados en una sesión.
- RSVP: SIP puede usarlo para reservar los recursos de red, como el ancho de banda.
- TLS: para proporcionar a SIP privacidad a la información de señalización.

Dependiendo de los requisitos de señalización y de la aplicación, SIP puede usar otros protocolos. SIP no necesita ninguno de los protocolos anteriores. En el futuro, si están disponibles mejores protocolos que realicen funciones similares, SIP será capaz de usarlos casi sin cambios. La mayoría de la información asociada a estos protocolos es transportada en el cuerpo del mensaje de SIP, por lo que no son interpretados.

8.11.3.4. Flujo de los mensajes en una red SIP

La siguiente figura muestra una red SIP básica, compuesta por proxies SIP y user agents con

conexión a la PSTN. Todos estos elementos están dentro de la misma red IP. El gateway SIP-PSTN enlaza ambas redes.

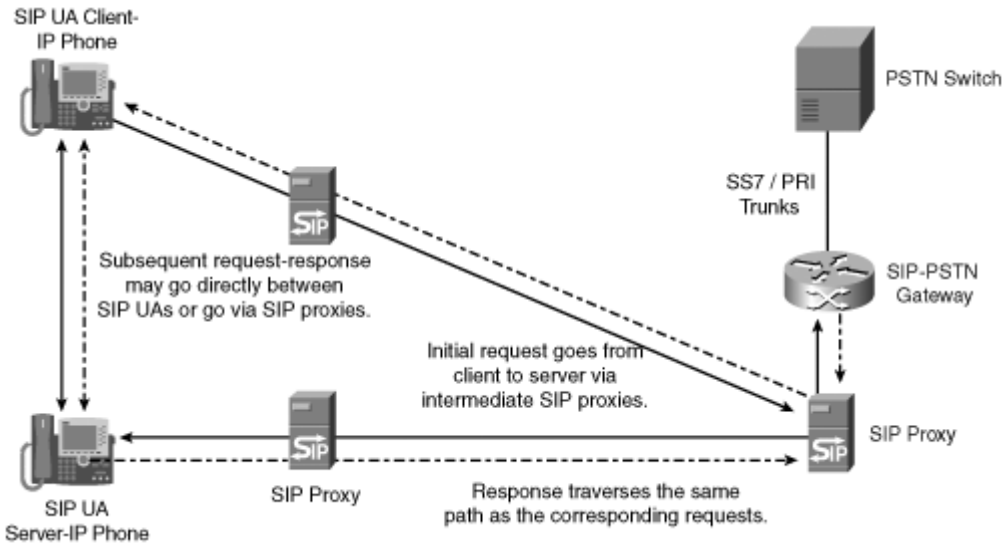


Ilustración 19: caminos de peticiones Request y Responde en una red SIP

Las líneas contínuas representan peticiones SIP, siendo las discontínuas las respuestas SIP.

8.11.3.5. Componentes de un mensaje SIP

Esta sección describe la estructura de los mensajes SIP, formación de direcciones y campos clave de la cabecera. Muchos de los mensajes SIP y de la sintaxis de la cabecera es igual a la de HTTP/1.1, estando descritos en la RFC 3261.

8.11.3.6. Direccionamiento SIP

Una dirección SIP identifica a un usuario o a un recurso dentro de una red. Las direcciones SIP son llamadas SIP URI, y tienen parecido a una dirección de e-mail:

sip:user@domain:port

sip:user@host:port

El campo usuario identifica a un nombre de usuario, o a un número de teléfono, dentro del contexto del dominio SIP. El puerto es un campo opcional, siendo por defecto el 5060.

A una dirección SIP pública de un usuario o un recurso se le llama Address-of-Record (AOR), y es una SIP URI que es enrutable globalmente y que apunta a un dominio cuyo servicio de localización mapea la AOR o otra SIP URI donde se encuentra el usuario. El puerto por defecto de una SIPS URI es el 5061.

8.11.3.7. Mensajes SIP

Los mensajes SIP pueden ser divididos en peticiones y respuestas, como se describe a continuación.

8.11.3.7.1. SIP Requests

Las peticiones SIP son mensajes que envían los clientes a los servidores para invocar operaciones SIP. La RFC 3261 define seis tipos de peticiones o métodos SIP que permitan a un User Agent y a un proxy a localizar usuarios e iniciar, modificar y abandonar sesiones:

- INVITE: invita al receptor a participar en una sesión, aunque también se puede usar esta petición para modificar las características de una sesión ya establecida.
- ACK: una petición ACK confirma que el UAC ha recibido la respuesta final a una petición de INVITE.
- OPTIONS: un UA usa una petición OPTIONS para preguntar a un User Agent Server por sus capacidades.
- BYE: un UA usa una petición BYE para pedir la terminación de una sesión previamente establecida.
- CANCEL: una petición de CANCEL permite a los UACs y a los servidores de red cancelar peticiones en marcha, como una INVITE.
- REGISTER: un cliente usa una petición REGISTER para registrar su localización correspondiente a su AOR.

8.11.3.7.2. SIP Responses

Un servidor envía una respuestas SIP a un cliente para indicar el estado de una petición SIP que el cliente previamente envió al servidor. Las respuestas SIP están numeradas de 100 a 699, agrupándose según el primer dígito.

Las respuestas SIP se clasifican como provisionales o finales. Una respuesta provisional indica el progreso por el servidor pero no indica el resultado final de la petición. Este tipo de respuestas están numeradas como 1xx. El resto de respuestas son finales:

- Las de tipo 2xx indican un procesamiento exítoso en la petición SIP.
- Las de tipo 3xx indican que la petición SIP necesita ser redirigida a otro UAS para su procesamiento.
- Las de tipo 4xx, 5xx o 6xx indican fallos en el procesamiento de las peticiones SIP.

Class of Response	Status Code	Explanation
Informational	100	Trying
	180	Ringing
	181	Call is being forwarded
	182	Queued
	183	Session progress
Success	200	OK
Redirection	300	Multiple choices
	301	Moved permanently
	302	Moved temporarily
	305	Use proxy
	380	Alternative service
Client-Error	400	Bad request
	401	Unauthorized
	402	Payment required
	403	Forbidden
	404	Not found
	405	Method not allowed
	406	Not acceptable
	407	Proxy authentication required
	408	Request timeout
	410	Gone
	413	Request entity too large
	414	Requested URL too large
	415	Unsupported media type
	416	Unsupported URI scheme
	420	Bad extension
	421	Extension required
	423	Interval too brief
	480	Temporarily not available
	481	Call leg or transaction does not exist
	482	Loop detected
	483	Too many hops
	484	Address incomplete
	485	Ambiguous
	486	Busy here
	487	Request terminated

Class of Response	Status Code	Explanation
	488	Not acceptable here
	491	Request pending
	493	Undecipherable
Server-Error	500	Internal server error
	501	Not implemented
	502	Bad gateway
	503	Service unavailable
	504	Server timeout
	505	SIP version not supported
	513	Message too large
Global Failure	600	Busy everywhere
	603	Decline
	604	Does not exist anywhere
	606	Not acceptable

Tabla 4: respuestas SIP

8.11.3.7.3. Estructura de un mensaje SIP

Un mensaje SIP consiste en lo siguiente:

- Una línea de inicio.
- Uno o más campos cabecera.
- Una línea vacía indicando el final de los campos cabecera.
- Un cuerpo de mensaje opcional.

Hay que terminar la línea de inicio, cada línea de la cabecera y la línea vacía por los caracteres CRLF.

La línea de inicio de una petición SIP es una Request-Line. La línea de inicio de una respuesta SIP es una Status-line.

Una Request-line indica el método SIP, la URI del emisor, y la versión SIP. Una Status-line describe la versión SIP, el código de respuesta SIP, y una frase opcional como explicación.

En la tabla 5 se muestra un ejemplo de petición SIP.

INVITE sip:bob@proxy.company.com SIP/2.0	**Request Line**
Via: SIP/2.0/UDP ph1.company.com:5060;branch=z9hG4bK83749.1	SIP Message headers
From: Alice <sip:alice@company.com>;tag=1234567	
To: Bob <sip:bob@proxy.company.com>	
Call-ID: 12345601@ph1.company.com	
CSeq: 1 INVITE	
Contact: <sip:alice@ph1.company.com>	
Content-Type: application/sdp	
Content-Length: ...	
	Blank line between SIP header fields and body
v=0	SDP body in SIP message
o=alice 2890844526 28908445456 IN IP4 172.18.193.102	
s=Session SDP	
c=IN IP4 172.18.193.102	
t=0 0	
m=audio 49170 RTP/AVP 0	
a=rtpmap:0 PCMU/8000	

Tabla 5: componentes de una petición SIP

La tabla 6 muestra la estructura de una respuesta SIP:

SIP/2.0 200 OK	**Status (Response) Line**
Via: SIP/2.0/UDP ph1.company.com:5060;branch=z9hG4bK83749.1	SIP message headers
From: Alice <sip:alice@company.com>;tag=1234567	
To: Bob <sip:bob@proxy.company.com>;tag=9345678	
Call-ID: 12345601@ph1.company.com	
CSeq: 1 INVITE	
Content-Length: ...	
	Blank line between SIP header fields and body
v=0	SDP body in 200 OK response
o=bob 3800844316 3760844696 IN IP4 172.18.193.109	
s=Session SDP	
c=IN IP4 172.18.193.109	
t=0 0	
m=audio 48140 RTP/AVP 0	
a=rtpmap:0 PCMU/8000	

Tabla 6: componentes de una respuesta SIP

Respecto a los campos cabecera, SIP sigue el mismo formato que el definido para las cabeceras HTTP, en la RFC 2616. Cada campo cabecera consiste en un nombre seguido por dos puntos, ":", y el valor del campo.

La siguiente tabla describe las funciones de las cabeceras SIP:

SIP Header	Explanation
From	Indica la identidad del iniciador de la petición SIP. Suele ser el AOR del emisor, seguido opcionalmente por el display name.
To	Indica el receptor de la petición SIP. Suele ser el AOR del receptor y opcionalmente el display name.
Call-ID	Esta cabecera identifica a una seria de mensajes SIP que formen parte de un diálogo.
Cseq	Esta cabecera está compuesta por un valor entero y un nombre de método. Identifica, ordena y secuencia peticiones SIP dentro de un diálogo.
Via	Indica el camino seguido por la petición, e identifica donde deben ser enviadas las respuestas.
Contact	Identifica una SIP URI o SIPS URI donde un UA quiere recibir una nueva petición SIP.
Allow	Indica el conjunto de métodos soportados por el UA que ha generado el mensaje.
Supported	Lista todas las extensiones SIP soportadas por el UA.
Require	Similar a Supported, indica las extensiones que deben ser soportadas para que se establezca la comunicación.
Content-Type	Indica el tipo de cuerpo de mensaje que está unido a una petición o respuesta SIP. Debe estar presente si el mensaje SIP tiene cuerpo.
Content-Length	Indica el tamaño del cuerpo en un mensaje SIP.

Tabla 7: Cabeceras del protocolo SIP

8.11.3.8.Transacciones y diálogos SIP

Una sesión de señalización entre dos user agents puede estar compuesta de una o más transacciones. Una transacción SIP ocurre entre un UAC y un UAS, los cuales pueden involucrar a uno o más servidores SIP intermediarios que actúen de proxy o redirigiendo. Una transacción SIP comprende todos los mensajes que empiezan con las peticiones SIP iniciadas desde un UAC, hasta la repuesta final, es decir, hasta una respuesta no 1xx, recibida desde el UAS.

Una transacción SIP se identifica por el Call-ID, via-branch, local tag, remote tag y el valor de Cseq. La figura 20 muestra una transacción SIP REGISTER entre un UAC y en servidor de registro. La transacción SIP comprende un mensaje de petición SIP seguido por uno o más mensajes de respuesta SIP. En este caso, el mensaje REGISTER es una petición del UAC a un servidor de registro. Las respuestas SIP son 100 Trying y 200 OK. El UAS envía respuestas al UAC indicando el estado de la petición SIP.

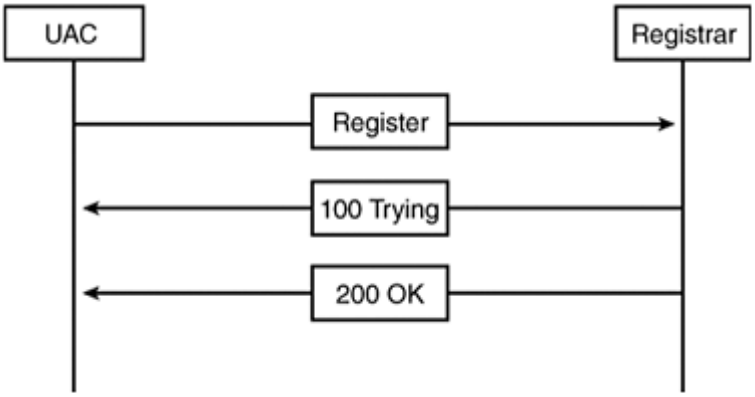


Ilustración 20: Transacción SIP REGISTER

Una transacción SIP puede acabar en el establecimiento, modificación o terminación de una sesión multimedia. El establecimiento de una sesión también acaba en una señalización de la relación SIP entre los extremos, llamada diálogo. Un diálogo es definido como una relación SIP punto a punto entre dos o más UA que persiste durante la duración de la sesión. Un diálogo identifica su estado mediando el Call-ID, el tag local y el tag remoto. Múltiples transacciones SIP puede ocurrir dentro del contexto de un diálogo SIP.

Una transacción INVITE-200 OK exitosa desemboca en el establecimiento de un diálogo SIP, y en una sesión de audio o vídeo entre los participantes. Después de que la sesión se establezca, se pueden intercambiar mensajes INVITE dentro del mismo diálogo usando el mismo Call-ID y los tags para modificar los parámetros de los medios de la sesión. Al final, se puede terminar la sesión usando la transacción BYE o transferirla a otro dispositivo usando la transacción REFER, dentro del mismo contexto de diálogo.

Las transacciones SIP usan igualmente protocolos orientados a conexión, como TCP o SCTP, o conexiones sin estado, como UDP. Para protocolos sin estado, SIP especifica que las aplicaciones SIP tengan timers para retransmitir las peticiones SIP, para intentar garantizar la entrega punto a punto.

8.12.Interconexión entre PSTN y VoIP

En el mundo real, las tecnologías VoIP y PSTN coexisten y seguirán haciéndolo en los próximos años, por lo que es necesario interconectarlas. Se describe una propuesta, denominada Cisco Packet Telephony, que persigue este fin.

Se organiza en tres planos lógicos: control de conexión, control de llamadas y servicios. Cada plano representa un aspecto distinto de un servicio de voz, e interactúa con otros planos lógicos a través de interfaces definidas. Los tres planos están organizados según una jerarquía, con el control de conexión en el nivel más bajo, el control de llamadas sobre el de de conexión, y los servicios sobre el control de llamadas. La figura siguiente muestra la composición funcional de la arquitectura:

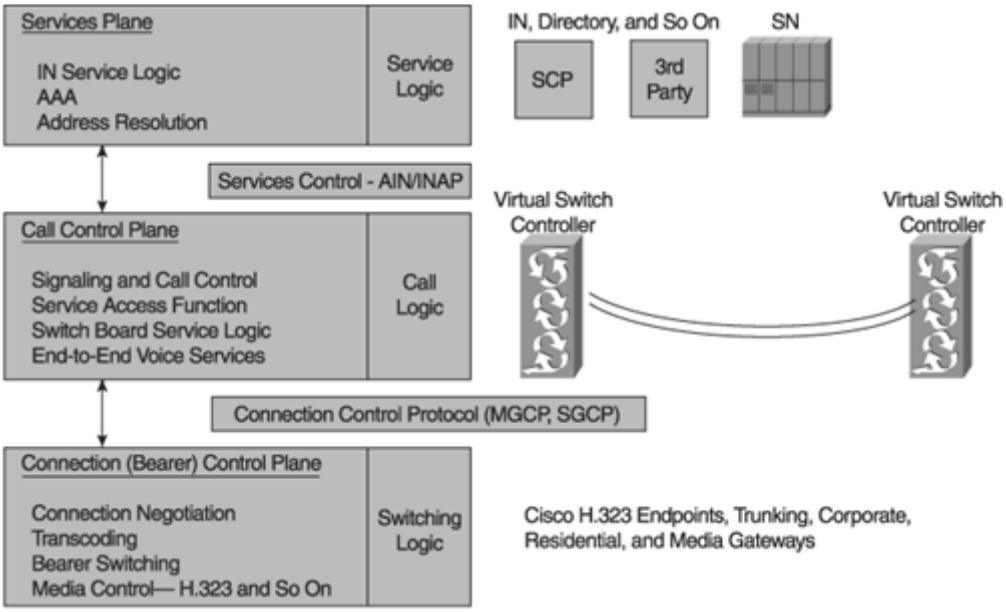


Ilustración 21: descripción de la arquitectura Cisco Packet Telephony

Una descripción de los planos es:

- El plano de control de conexión proporciona la funcionalidad que es necesaria para configurar, mantener y cerrar un camino de voz a través de una red de paquetes. Ejemplos de gateways de medios (MG) que realizan funciones de control son Cisco AS5850, MGX8850, 2600, 3600 y 3810. El plano de control de conexión se comunica con el de llamadas usando un protocolo de control como el Media Gateway Control Protocol (MGCP).
- El plano de control de llamadas comprende la funcionalidad necesaria para señalizar, procesar y enrutar llamadas de voz y datos sobre la red de paquetes. Las funciones dentro de este nivel están cerca de aquellas encontradas en la lógica de procesamiento de llamadas de un switch de multiplexión en el tiempo (TDM). Funciones típicas del control de llamadas incluye el procesamiento del protocolo de señalización Signaling System 7 (SS7), manipulación y análisis de dígitos, selección de rutas e interfaz con programas de servicio externos.
- El plano de servicios comprende la lógica necesaria para proveer funciones como traducción de números, autenticación para tarificación, validación de tarjeta de crédito, VPN, etc.

A medida que aumentan las necesidades de interconexión entre diferentes redes de VoIP, los proveedores de servicio y portadores se enfrentan a nuevos retos en la configuración de redes soportando diferentes protocolos y elementos de red. Si además se añade los retos relacionados con la facturación y la seguridad, la complejidad se incrementa. La tabla siguiente muestra alguna de las tendencias en la interconexión de redes:

Moving From	Moving To
Flat bandwidth billing	Usage billing
Best effort	Service-level agreements (SLA) and QoS
Open network connections	Topology hiding
TDM interconnects	IP interconnects, signaling protocol interoperability
Same codec interconnects	Transcoding
Assumed trust	Security
Simple IP domain	Interconnected IP domains
Lawful intercept (LI) ability—optional	LI required

Tabla 8: tendencias en la interconexión VoIP

8.13. Aplicaciones y servicios sobre VoIP de Proveedores de Servicio

A finales de 2004, la compañía In-Stat estimaba que había 1,3 millones de líneas IP en conexiones de banda ancha en los Estados Unidos, con un crecimiento que alcanzaría los 3,9 millones al final de 2005. La mayoría de los proveedores de servicio de todo el mundo están de acuerdo en que IP ha ganado el nivel de transporte en la red para la voz, los datos, el video y los servicios de valor añanido.

Los proveedores de servicio tradicionalmente han sido clasificados como compañías de cable, de telecomunicaciones, y compañías de banda ancha y wireless. Con VoIP y sus servicios avanzados, una nueva clase de proveedores utiliza la infraestructura de red de los proveedores tradicionales y ofrece servicios de valor añadido (por ejemplo, AOL, Yahoo!, Skype y Google) compitiendo ahora por el mercado VoIP.

Cuando una organización planea consolidar sus redes de voz y datos en una red multiservicio, la aplicación inicial que se considera es toll-bypass, que permite enviar las llamadas de voz y fax entre las sedes a través de la red TCP/IP que ya se tenga para datos, evitando así las llamadas a larga distancia de la red PSTN, lo que se traduce en un ahorra inmediato de teléfono así como un mejor aprovechamiento de la red de datos. Normalmente, el retorno de la inversión (ROI) de toll-bypass se produce entre los tres y seis meses.

Mientras que la organización se siente cada vez más confortable con VoIP y toll-bypass, las siguientes aplicaciones que se consideran suelen ser las que faciliten el servicio al cliente, grupos de trabajo interactivos y formación a distancia. Ejemplos de aplicaciones de estas áreas son:

- Microsoft NetMeeting: cliente VoIP ya veterano, todavía se usa en muchos sitios, provee integración entre compartición de aplicaciones y video conferencia basada en H.323.
- Cisco IP Phone: tiene la apariencia de un terminal tradicional, pero con la funcionalidad de la telefonía IP. En lugar de conectarse a una PBX tradicional, trabaja en conjunción con una PBX basada en IP, como Asterisk. Estas PBXs no sólo proporcionan la misma funcionalidad que las tradicionales, sino que también toman ventaja de todos los servicios IP disponibles en la red.

- Softphone: extiende la funcionalidad de un terminal a un ordenador personal con un interfaz gráfico, que proporciona la misma funcionalidad que el terminal tradicional y además integra otros servicios como navegación Web, NetMeeting, etc.
- Clientes de mensajería con VoIP integrada: la inclusión de funcionalidades de VoIP en clientes de mensajería es algo habitual. GoogleTalk, Skype, Yahoo!, Gizmo y AOL son sólo algunas de las últimas aplicaciones software que integran esta funcionalidad.

Estos servicios se pueden considerar de primera generación. Para implementar otros servicios más avanzados, es necesario que los proveedores de servicios vayan adoptando estándares para permitir la conexión de la diferentes islas VoIP, así como la facturación y la gestión de llamadas.

Las necesidades de las organizaciones aumentan con el tiempo. Por ejemplo, cada vez más trabajadores realizan teletrabajo o están constantemente de un sitio para otro. La organización debe tener una infraestructura que haga transparente que las oficinas sean remotas, o se trabaje desde casa, debe derribar las barreas geográficas y de productividad. Se debe poder hacer o recibir llamadas, acceder a los servicios de mensajería, colaboración, email, indicar disponibilidad para la comunicación, todo ello sin tener en cuenta si es un teléfono regular, un softphone o un móvil.

Las organizaciones están incrementando el uso de las aplicaciones de conferencias de voz y video, compartición de documentos y herramientas de colaboración para reducir el tiempo y coste de los viajes.

Otra aplicación interesante es la de información de presencia, que consiste en que el receptor de una llamada indique su disponibilidad para ser llamado o para tratar temas laborales, evitando así llamadas improductivas.

Uso de teléfonos IP Wi-Fi habilita a los usuarios a conectarse desde cualquier sitio dentro de la organización, desde hotspots públicos y desde sus casas. Los teléfonos Wi-Fi permiten a los usuarios ser llamados a su extensión sin estar en sus mesas.

Algunos teléfonos Wi-Fi permiten al usuario conmutar entre la red VoIP y la red celular. Una persona puede iniciar una llamada VoIP desde su oficina, y conforme camine fuera de ella, la llamada será conmutada sin cortes a la red celular. La ventaja es la gran flexibilidad en las comunicaciones móviles, así como un gran ahorro en minutos de llamadas desde celulares que se harían desde la red de la empresa.

Otra mejora de telefonía VoIP consiste en la calidad de la voz. Al inicio de VoIP, las implementaciones trataban de dar la misma calidad que la tradicional PSTN, cuyo ancho de banda es de 4 kHz. Sin embargo, VoIP permite el uso de codecs con más ancho de banda, que samplean a 16 kHz, permitiendo la transmisión de frecuencias de hasta 8 kHz. Estos codecs ofrecen mucho mejor sonido sin una gran demanda de ancho de banda.

Por ejemplo, G722.2 es un codec de banda ancha definido por ITU-T. Tiene un bit rate de entre 6 y 23,85 kbps, y toma samples a 16kHz.

Con codecs así, se puede plantear la implantación de aprendizaje a distancia, entrenamiento a demanda, conferencias y sesiones multimedia, ya que la experiencia de usuario es mucho mejor.

9. Asterisk

9.1. Introducción a Asterisk

Asterisk es un programa de software libre, con licencia GPL, que implementa una central telefónica completa (PBX). Como tal, se le puede conectar una serie de teléfonos para llamarse entre sí, o bien conectarlo con un proveedor externo, ya sea mediante conexión VoIP o digital, tanto básicos como primarios.

El creador de Asterisk fue Mark Spencer, que sigue siendo su principal desarrollador, aunque ya hay otros programadores que contribuyen con nuevas funcionalidades y arreglando errores. Aunque fue desarrollado para Linux, actualmente funciona para los sistemas operativos BSD, Mac OS X, Solaris y Windows.

Asterisk proporciona muchas de las funcionalidades que para tenerlas antes era necesario contar con costosos sistemas propietarios PBX, como buzón de voz, conferencias, IVR, distribución automática de llamadas, etc. Además, cuenta con un lenguaje de script para crear nuevas funcionalidades en el dialplan.

Para conectar Asterisk a líneas analógicas individuales o en trunks, son necesarias tarjetas electrónicas telefónicas FXS o FXO fabricadas por Digium u otros proveedores, que proporciona conexión con teléfonos, líneas de teléfono, líneas T1 y E1, y otros servicios telefónicos analógicos y digitales.

Asterisk soporta un amplio rango de protocolos relativos a vídeo y VoIP, incluyendo SIP, Media Gateway Control Protocol (MGCP) y H.323. Puede interactuar con los terminales SIP tanto para registrarse como de gateway con redes PSTN. Ha desarrollado un protocolo, Inter-Asterisk eXchange (IAX2), que proporciona trunking de llamadas entre PBXs Asterisk, configuración distribuida, etc. Ya hay teléfonos que soportan directamente el protocolo IAX2.

Además de protocolos para VoIP, Asterisk soporta muchos protocolos de conmutación de circuitos tradicionales como ISDN y SS7. Este soporte requiere tarjetas hardware apropiadas. Cada protocolo requiere la instalación de módulos software como Zaptel, Libpri, Libss7, chanss7, wanpipe y otros.

A continuación se describen algunas de las características más relevantes de Asterisk:

- Contestación automática de llamadas: se puede configurar un contestador automático que ayude con las llamadas entrantes contestándolas automáticamente. La operadora virtual responde a los dígitos que la persona que llama va marcando, enrutando las llamadas a extensiones específicas, proporcionando acceso a información pregrabada, recibiendo un mensaje de voz u otras acciones.
- Transferencia de llamadas: se puede transferir llamadas de una extensión a otra mediante una transferencia atendida o desatendida.
- Opción de no molestar: permite a cualquier usuario configurar su extensión para que no reciba llamadas por un periodo determinado.
- Parqueo de llamadas: permite aparcar una llamada un tiempo predefinido antes de volver a atenderla desde otra extensión.

- Captura de llamadas: permite coger una llamada que estaba sonando en otra extensión.
- Monitorización y grabación de llamadas: a través de un código predeterminado se ejecuta un comando que permite escuchar la conversación mantenida en una extensión. Adicionalmente, se permite grabar las conversaciones de cualquier extensión de forma aleatoria o programada.
- Buzón de voz: permite escuchar mensajes dejados por llamadas que no pudieron ser atendidas.
- Conferencias: cada extensión tiene asociado un cuarto para conferencias, que podrá ser usado por otros usuarios, ya sea internos o externos.
- Informes de llamadas: Asterisk genera registro de detalle de llamadas o CDRs (Call Detail Records), pudiéndolos almacenar en una base de datos.
- Colas de atención de llamadas: permite que un número ilimitado de llamantes pueda permanecer en espera hasta que haya recursos para atender la llamada.
- Llamada en espera: permite que se pueda interrumpir temporalmente una llamada para atender otra llamada, pudiendo acordar un tiempo para devolver o atender la primera llamada.
- Identificador de llamante: es una señal que se recibe entre las señales de RING o durante el proceso de establecimiento de la llamada, antes de ser contestada. A nivel IP está soportada, pero a nivel PSTN es el proveedor de telefonía quien debe habilitarla.
- Bloqueo de número: permite ocultar el número con el que se llama desde Asterisk. No todas las redes de telefonía pública soportan esta característica.
- Envío y recepción de fax: Asterisk detecta automáticamente cuando se está recibiendo un fax, pudiendo ser enviado a una cuenta de correo para su revisión. Asterisk también puede ser configurado como servidor de fax, para que los documentos que se envíen a una cuenta de correo específica sean enviados como faxes.
- Listado interactivo del directorio de extensiones: Asterisk puede tener en su base de datos el directorio telefónico de la organización y permitir, por ejemplo, que la persona que llama pueda teclear los números correspondientes a las cuatro primeras letras del apellido para hablar con ella.
- Interactive Voice Response (IVR): proporciona opciones de respuesta por voz ante la recepción de llamadas. Se pueden crear menús de activación por dígitos o por voz para ejecutar comandos de cualquier tipo.
- Música en espera

9.2. Introducción a Elastix

Elastix es una distribución de Linux, completamente software libre, que integra un Servidor de Comunicaciones Unificadas con las siguientes características:

- PBX, mediante Asterisk.
- Fax, usando Hylafax.

- Mensajería Instantánea, mediante OpenFire.
- Email, mediante Postfix.
- Colaboración.

Elastix se basa en la distribución de Linux CentOS, orientada a servidores, bien conocida por ser un clon de RedHat y por su calidad. Sobre esta distribución instala y configura Elastix todo el software de comunicaciones, quedando listo para producción, a falta sólo de la configuración, al final de la instalación.
Los cuatro programas sobre los que pasa Elastix su funcionalidad, Asterisk, Hylafax, Openfire y Postfix, son programas líderes, cada uno en su campo, en el software libre.

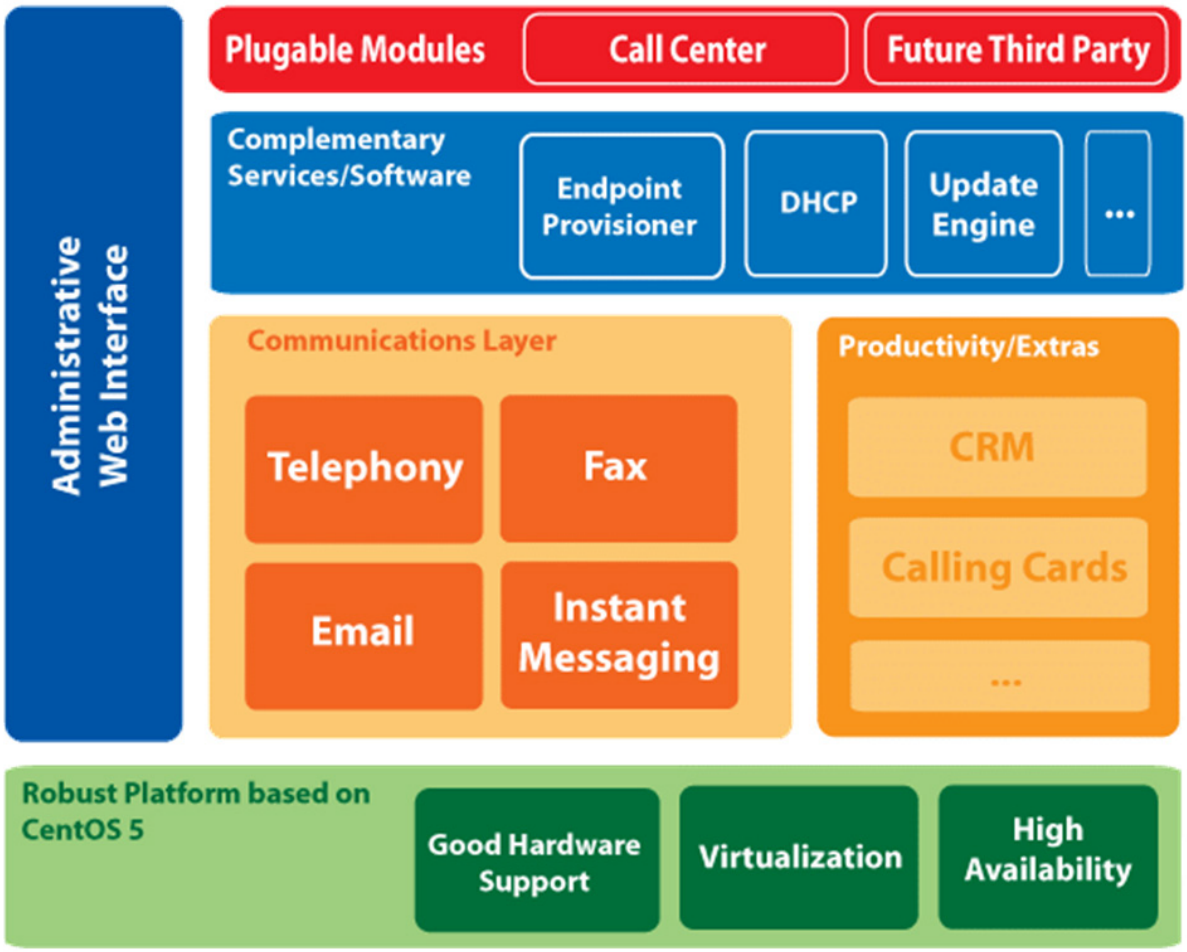


Ilustración 22: Esquema general de los componentes de Elastix

Elastix fue creado por la empresa ecuatoriana PaloSanto Solutions, que sigue siendo la responsable de su mantenimiento. Su primera versión fue en diciembre del 2006, y ya integraba muchas de las herramientas, ya administrables mediante una interfaz web.

9.3. Características de Elastix

9.3.1. VoIP PBX

- Grabación de llamadas con interface vía Web

- Voicemails con soporte para notificaciones por email
- IVR configurable y bastante flexible
- Soporte para sintetización de voz
- Herramienta para crear lotes de extensiones
- Cancelador de eco integrado
- Provisionador de teléfonos vía Web
- Soporte para videoteléfonos
- Interface de detección de hardware de telefonía
- Servidor DHCP para asignación dinámica de IPs a IP-Phones
- Panel de operador. Se puede ver toda la actividad telefónica de manera gráfica y realizar sencillas acciones drag-n-drop
- Parqueo de llamadas
- Reporte de detalle de llamadas (CDRs) con soporte para búsquedas por fecha, extensión y otros criterios
- Tarifación con reportación de consumo por destino
- Reporte de uso de canales por tecnología (SIP, ZAP, IAX, Local, H323)
- Soporte para colas de llamadas
- Centro de conferencias, desde donde se pueden programar conferencias estáticas o temporales
- Soporta protocolo SIP, IAX, H323, MGCP y SKINNY, entre otros
- Codecs soportados: ADPCM, G.711 (A-Law & μ-Law), G.722, G.723.1 (pass through), G.726, G.729 (si se compra licencia comercial), GSM, iLBC
- Soporte para interfaces analógicas FXS/FXO
- Soporte para interfaces digitales E1/T1/J1 a través de protocolos PRI/BRI/R2
- Soporte para interfaces bluetooth para celulares (canal chan_mobile)
- Identificación de llamadas
- Troncalización
- Rutas entrantes y salientes, las cuales se pueden configurar por coincidencia de patrones de marcado
- Soporte para follow-me
- Soporte para grupos de ringado
- Soporte para paging e intercom. El modelo de teléfono debe soportar también esta característica
- Soporte para condiciones de tiempo. Es decir que la central se comporte de un modo diferente dependiendo del horario
- Soporte para PINes de seguridad
- Soporte DISA
- Soporte Callback
- Editor Web de archivos de configuración de Asterisk
- Acceso interactivo desde el Web a la consola de Asterisk

9.3.2. Fax

- Servidor de Fax administrable desde Web
- Visor de Faxes integrado, pudiendo descargarse los faxes desde Web en formato PDF.

- Aplicación fax-a-email
- Personalización de faxes-a-email
- Control de acceso para clientes de fax
- Puede ser integrado con WinprintHylafax. Esta aplicación permite, desde cualquier aplicación Windows, enviar a imprimir un documento y éste realmente se envía por fax.
- Configurador Web de plantillas de e-mails

9.3.3. Email

- Servidor de Email con soporte multi-dominio
- Administrable desde Web
- Interface de configuración de dominios de Relay
- Cliente de e-mail basado en Web
- Soporte para cuotas configurable desde el Web

9.3.4. Colaboración

- Calendario integrado con PBX, con soporte para recordatorios de voz
- Libreta telefónica con capacidad click-to-call

9.3.5. Mensajería instantánea

- Servidor de mensajería instantánea basado en OpenFire e integrado a PBX con soporte para protocolo Jabber, lo que permite usar una amplia gama de clientes de IM disponibles
- Se puede iniciar una llamada desde el cliente de mensajería si se usa el cliente Spark
- El servidor de mensajería es configurable vía Web
- Soporta grupos de usuarios
- Soporta conexión a otras redes de mensajería como MSN, Yahoo Messenger, GTalk, ICQ, etc.
- Reporte de sesiones de usuarios
- Soporte para plugins
- Soporta LDAP
- Soporta conexiones server-to-server para compartir usuarios

9.3.6. General

- Ayuda en línea embebida
- Traducido a más de 20 idiomas
- Monitor de recursos del sistema
- Configurador de parámetros de red
- Control de apagado/reinicio de la central vía Web
- Manejo centralizado de usuarios y perfiles gracias al soporte de ACLs
- Administración centralizada de actualizaciones
- Soporte para backup/restore a través del Web
- Soporte para skins

- Interface para configurar fecha/hora/uso horario de la central

9.3.7. Extras

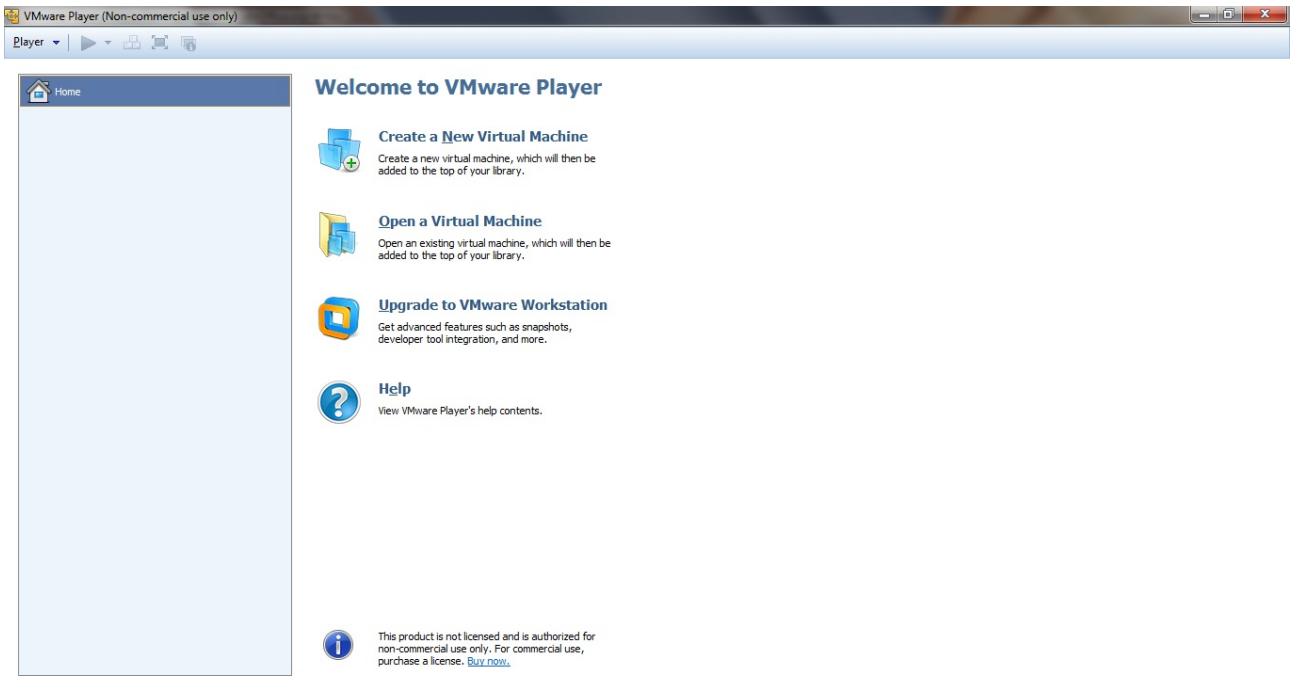
- Interface de generación de tarjetas de telefonía basada en software A2Billing
- CRM completo basado en el producto vTigerCRM
- Versión open source de SugarCRM
- Módulo de call center con marcador predictivo incluido

9.4. Instalación de Elastix

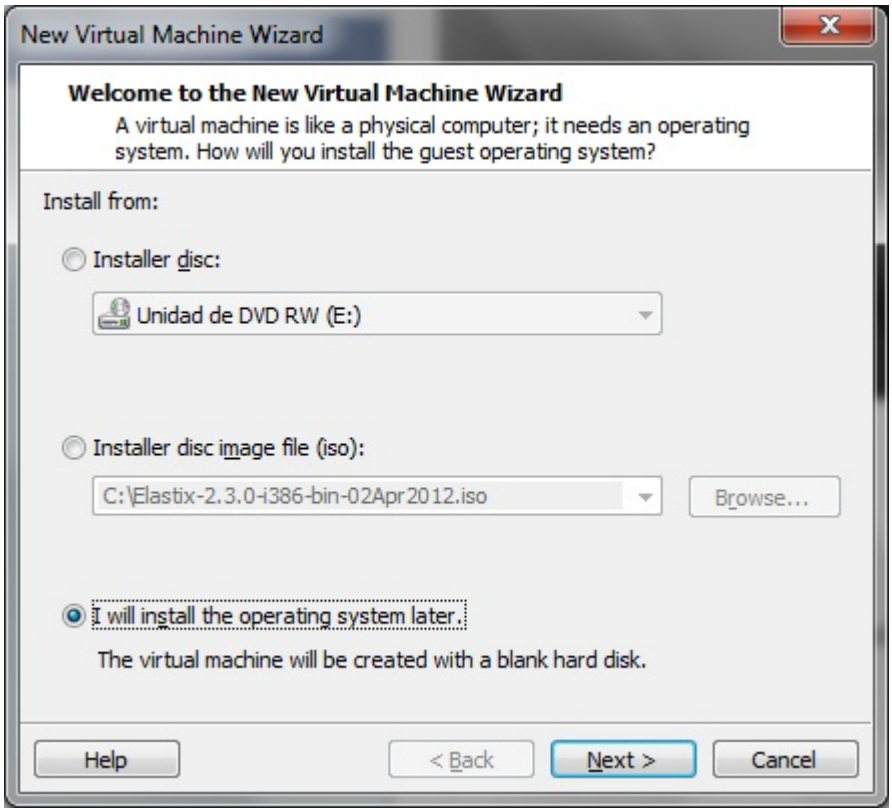
La instalación de Elastix se ha realizado bajo el software de virtualización Vmware Player, versión 5.0.1 build-894247, descargable desde http://www.vmware.com/go/downloadplayer/.

La versión de Elastix instalada es la 2.3.0, la última estable a la hora de realizar este PFC, descargable desde http://www.elastix.org/index.php/es/descargas/distro-principal.html, en la versión de 32 bits. La descarga consiste en un archivo .iso, de nombre Elastix-2.3.0-i386-bin-02Apr2012.iso, en el que está la distribución con todo el software necesario.

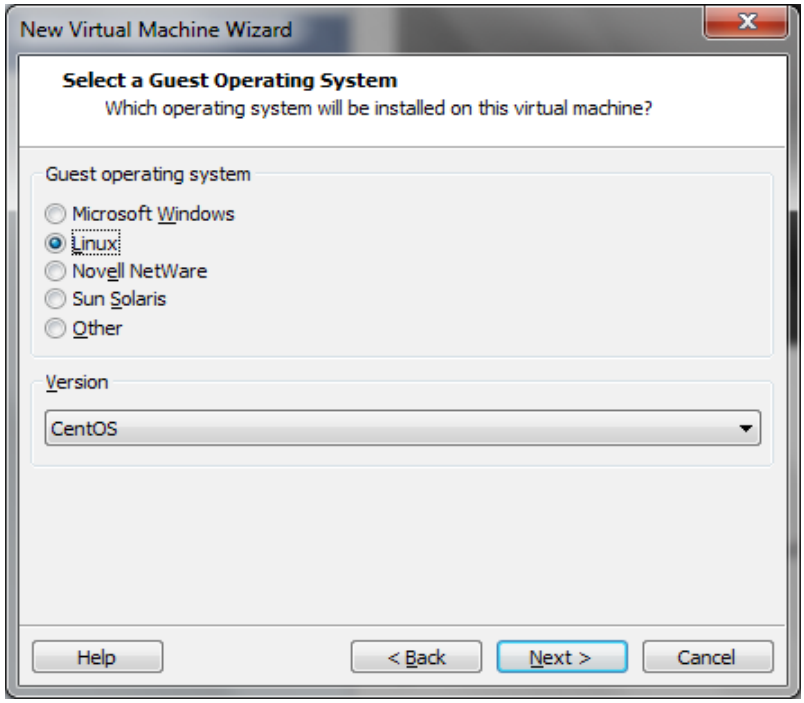
Para instalarlo debemos, como es lógico, crear una máquina virtual en Vmware Player. Para ello, se abre el programa y se selecciona la opción "Create a New Virtual Machine":



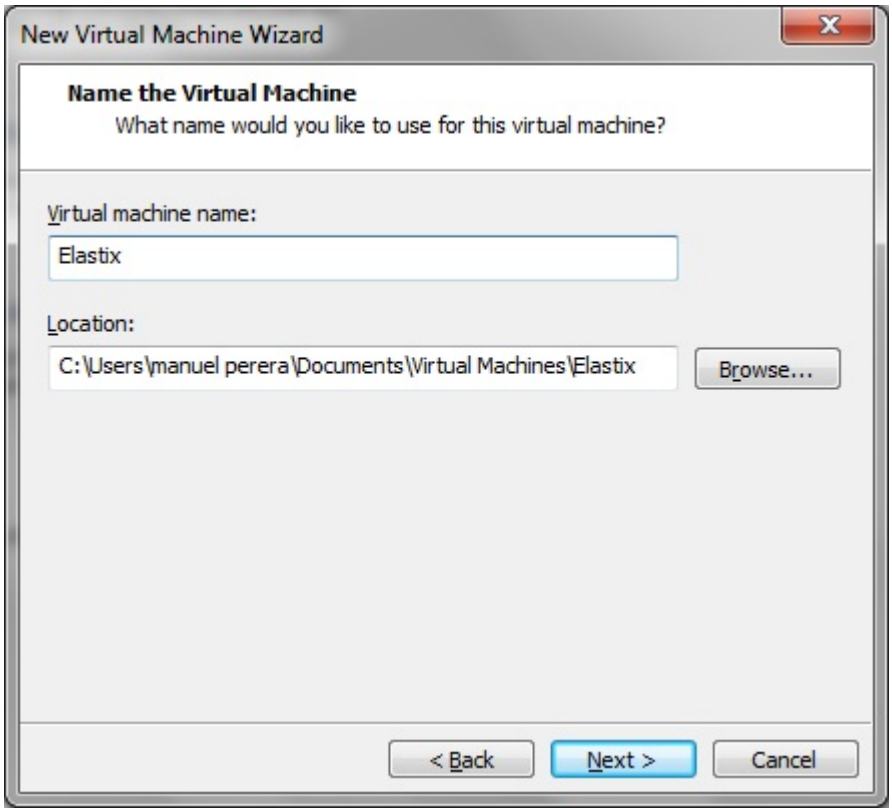
A continuación se inicia el asistente de creación de una máquina virtual, preguntando donde está el sistema operativo de la máquina que queremos crear. De momento, se crea solamente la máquina virtual sin instalar el sistema operativo, por lo que elije la opción "I will install the operating system later":



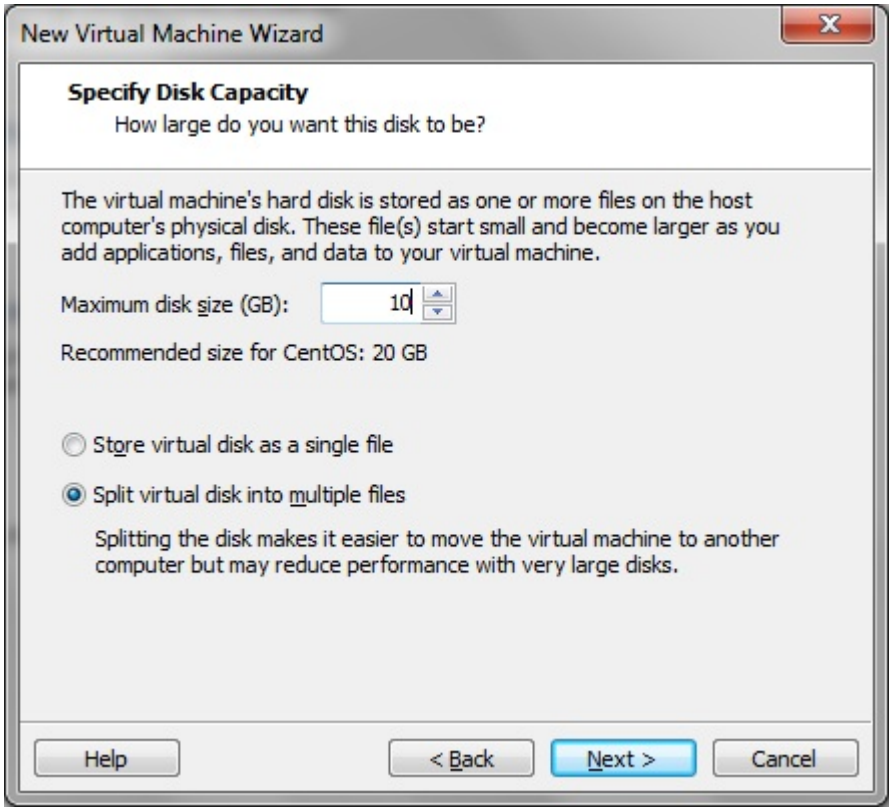
Después de pulsar "Next>" se nos pide el tipo de sistema operativo que se va a instalar. Se selecciona Linux, versión CentOS:



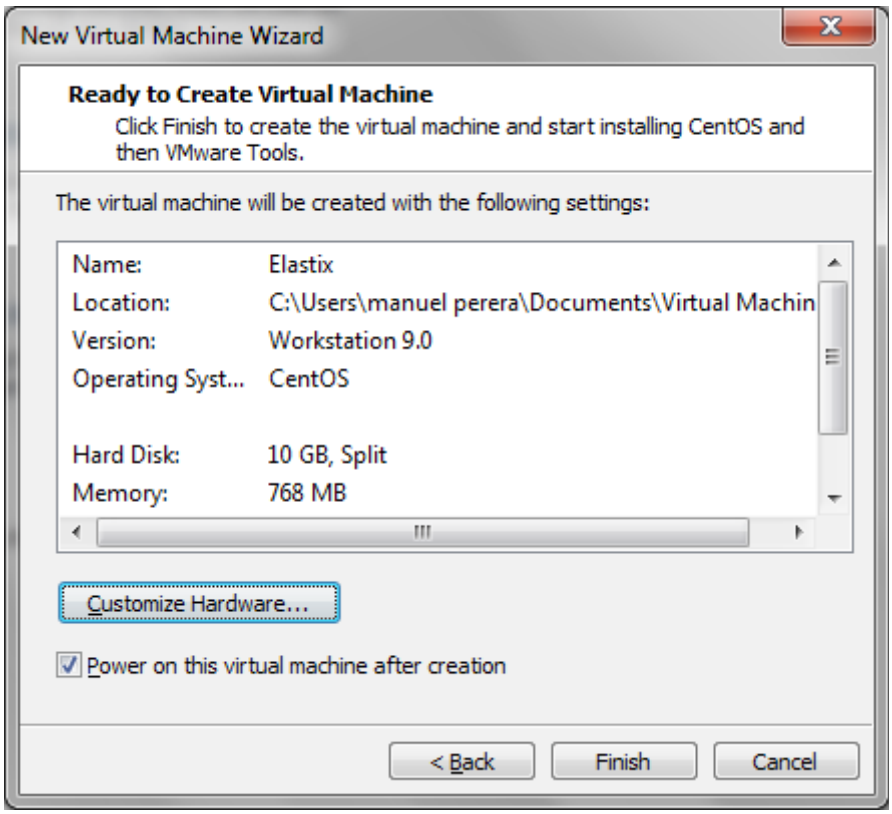
A continuación se indica el nombre de la máquina virtual y su ubicación en el disco duro:



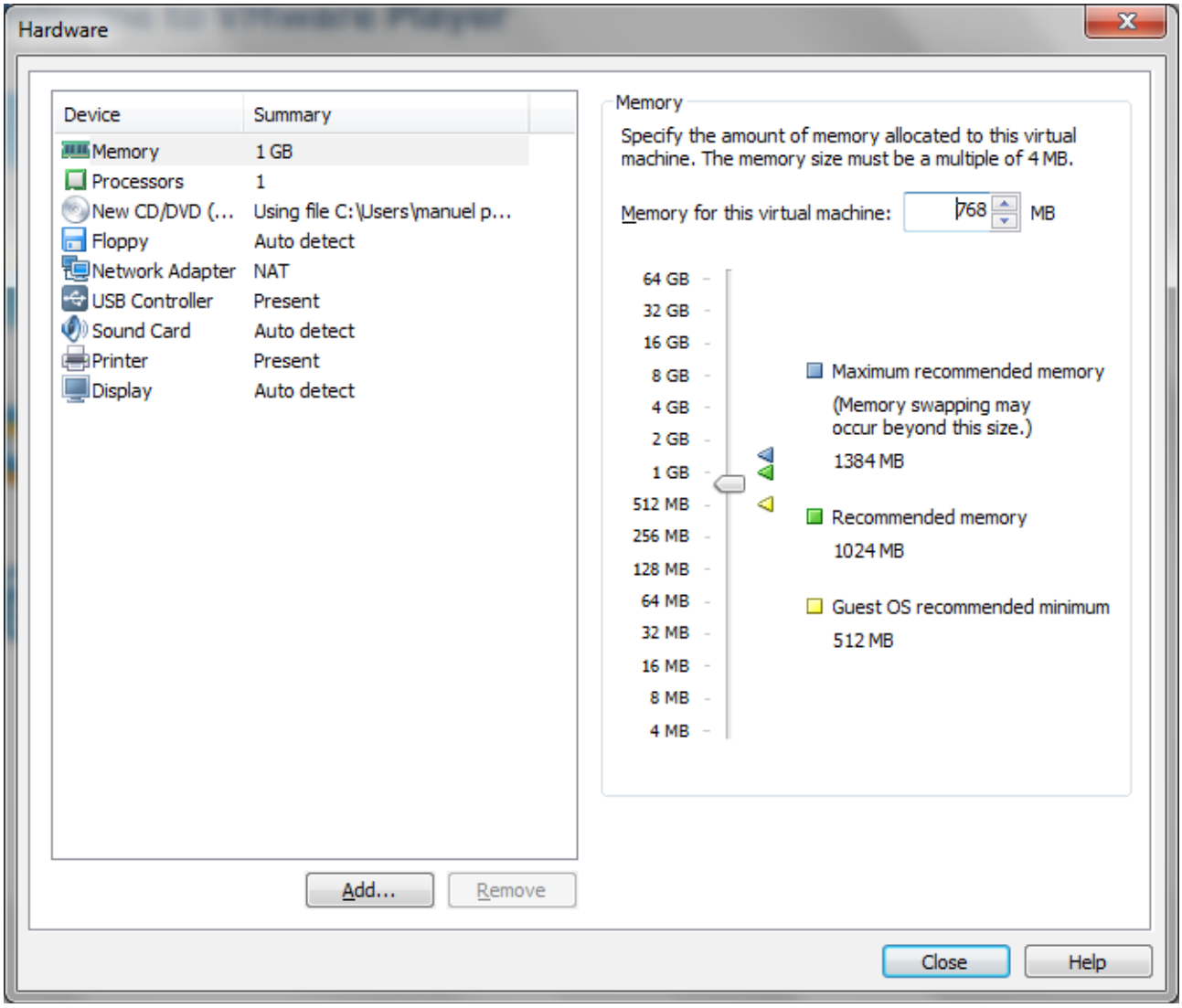
Seguidamente se indica el tamaño del disco virtual (10 GB), y si se implanta en un solo disco (single file) o dividido en varios (multiples files). Se elige la opción de multiples ficheros:



En la pantalla siguiente, antes de pulsar "Finish" para que se cree la máquina virtual, pulsamos sobre "Customize Hardware...":



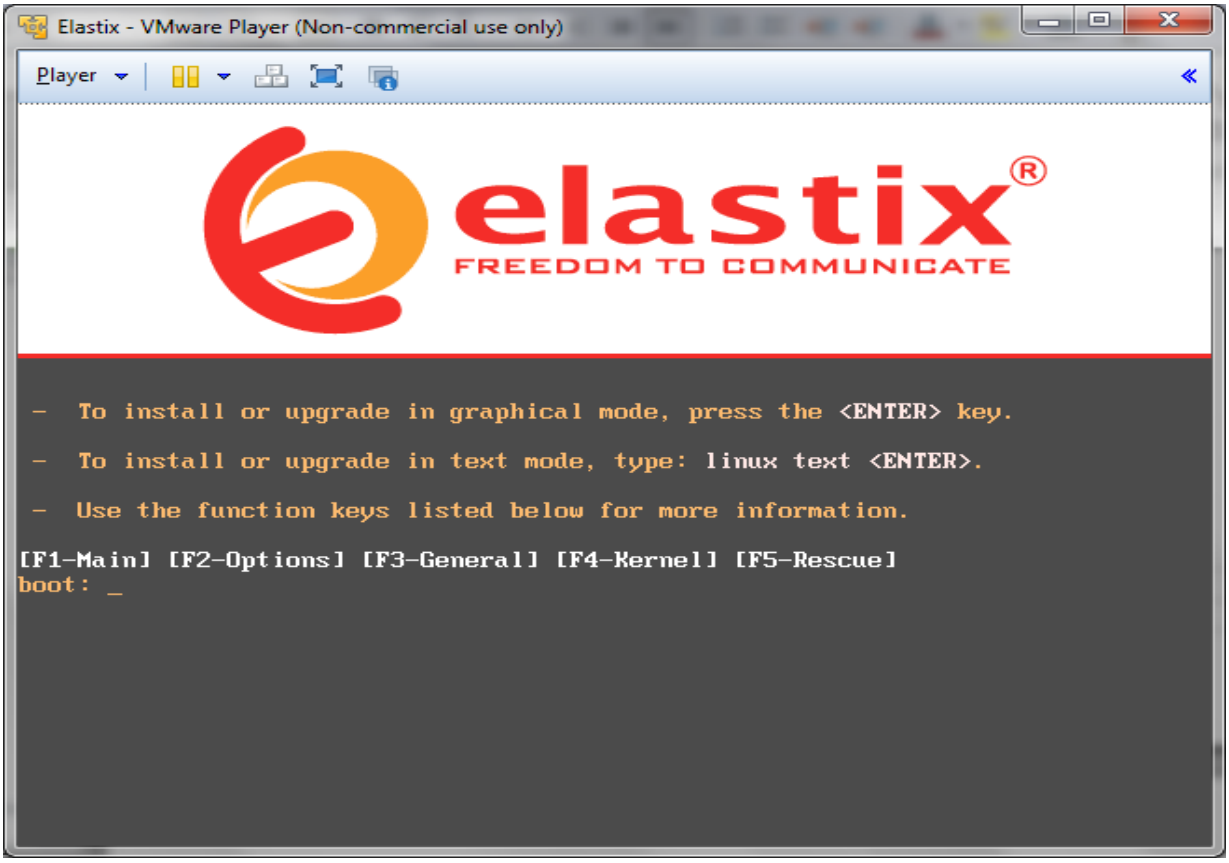
En la pantalla de configuración de hardware, indicamos una memoria para la máquina virtual de 768 MB:



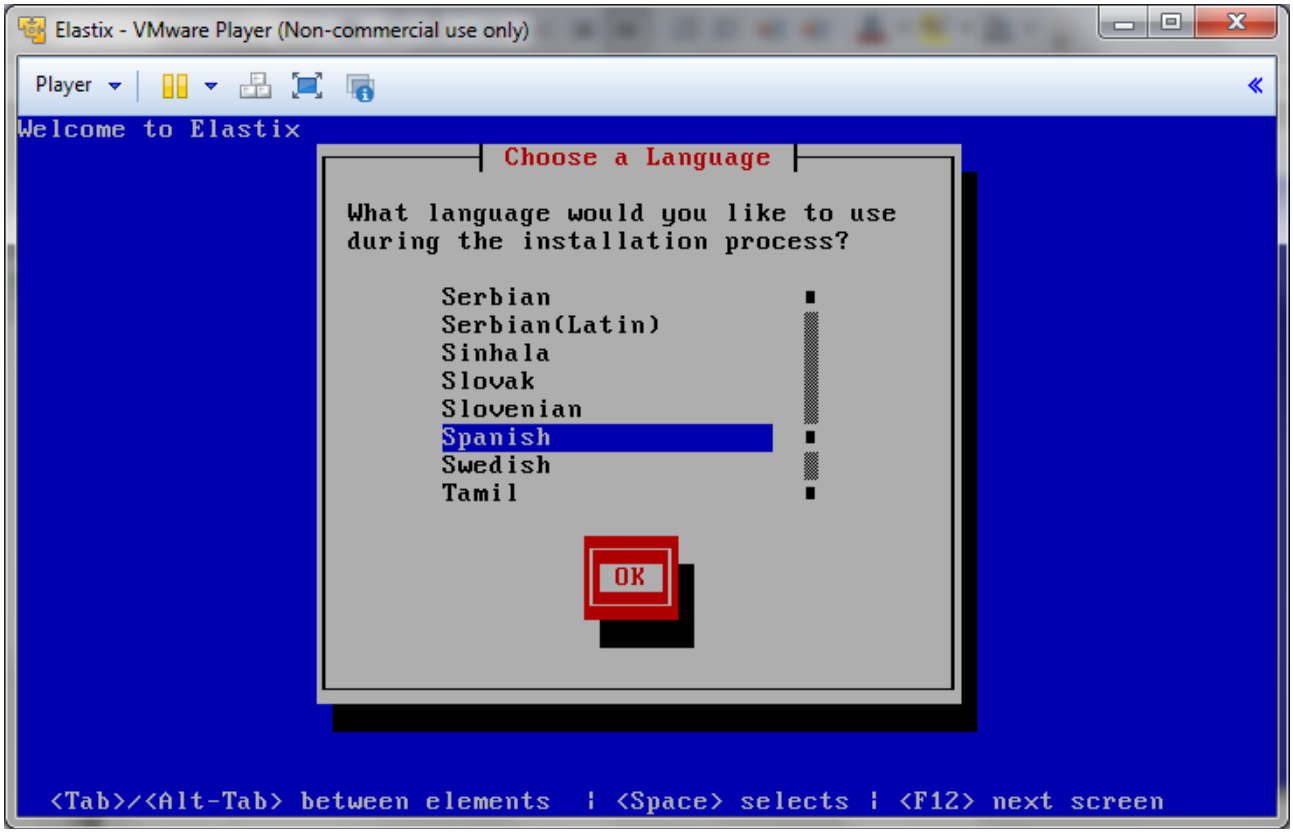
Además, se edita el CD/DVD de la máquina virtual y se le mapea el archivo .iso de Elastix. Para ello, en la opción "Use image file:" se selecciona dicho archivo. De esta forma, cuando se arranque la máquina virtual se tendrá el contenido de la distribución en el CD.

Pulsamos "Close" para cerrar esta ventana, y "Finish" en el asistente de creación de la máquina virtual para que, ahora sí, se proceda con la creación.

Una vez creada, se selecciona y se pulsa "Play virtual machine", con lo que se inicia la máquina virtual. Al tener conectado el CD, sale la pantalla de inicio de Elastix:

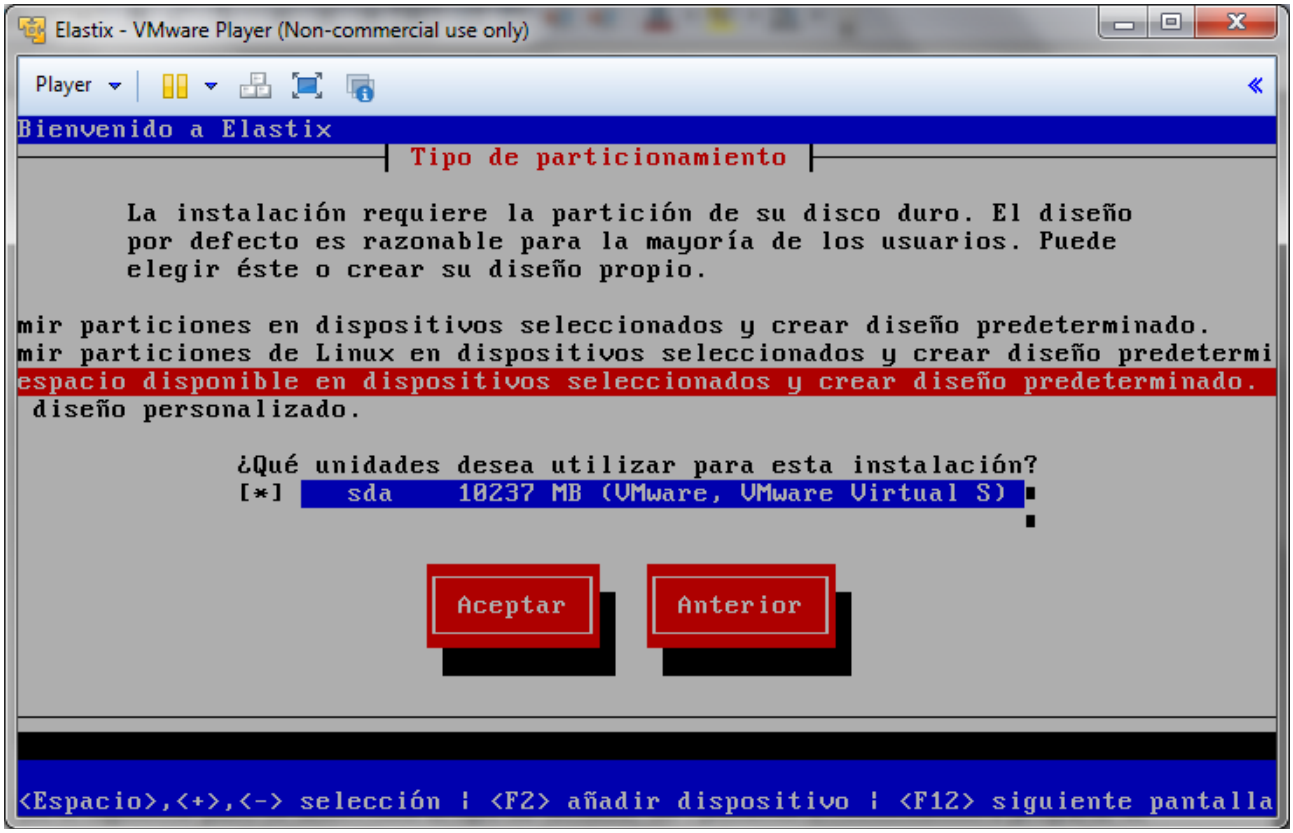


Después de pulsar Intro, sale la pantalla de selección del idioma de la instalación, en este caso español:



En el siguiente paso se selecciona el tipo de teclado, "es", pasando a continuación a preparar el particionamiento del disco. Al ser una instalación nueva, el asistente nos pregunta si se desea inicializar la unidad, eliminando todos los datos, a lo que respondemos afirmativamente.

En el siguiente paso le decimos al instalador que utilice todo el espacio disponible en el disco virtual, y que cree el particionamiento que tenga predeterminado.



Al no ser una instalación para poner en producción, no nos metemos en esos detalles, por lo que a la pregunta de si deseamos revisar y modificar la capa de particiones respondemos que no.

A continuación, se configura la intefaz de red, eth0, con los siguientes valores:

- Activar al inicio.
- Activar soporte Ipv4.
- Configuración manual TCP/IP.
- Dirección IP: 192.168.0.101
- Máscara de red: 255.255.255.0
- Puerta de enlace 192.168.0.1
- Dns primario: 192.168.0.1
- Nombre del host: Elastix

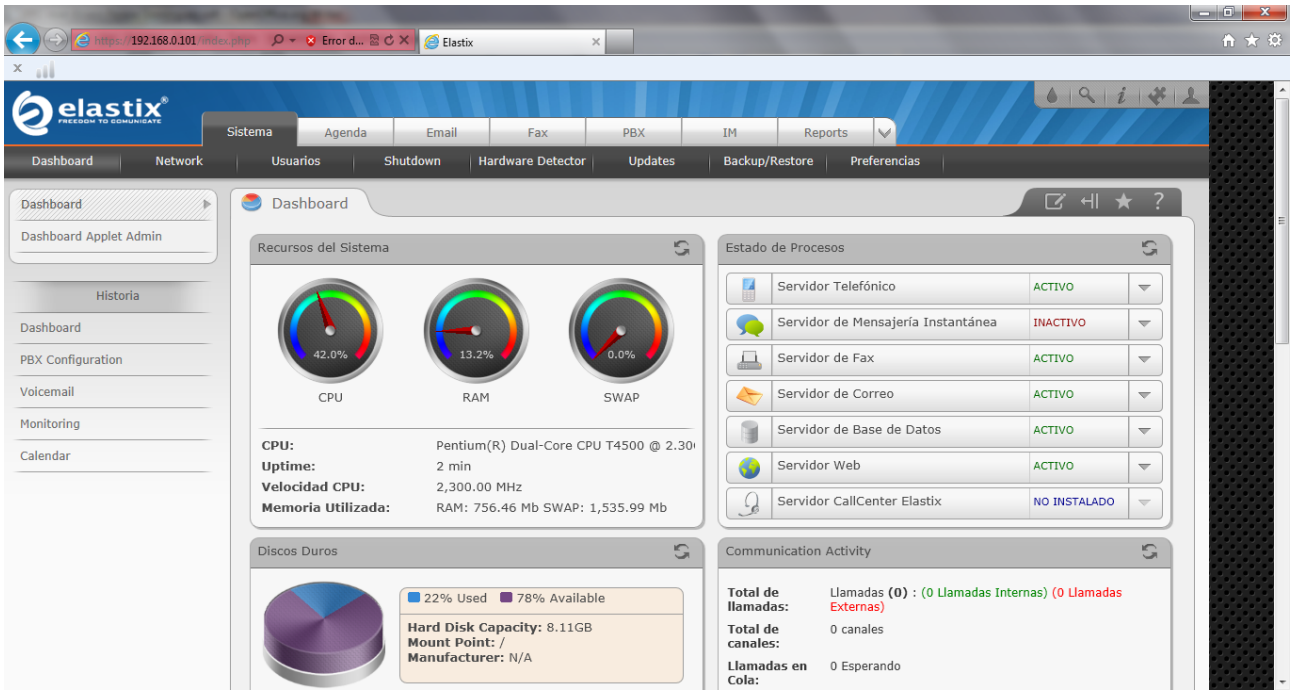
Para finalizar, se indica que la zona horaria es la de Europa/Madrid, y se introduce la contraseña de root.

Seguidamente comienza la instalación, que tarda pocos minutos, produciéndose un reinicio al acabar. En este primer inicio, Elastix pregunta por las contraseñas de root de la base de datos MySql, y por la del usuario admin para configurar los componentes de Elastix. Una vez finalizado todo el proceso, tendremos todos los componentes instalados y listos para ser

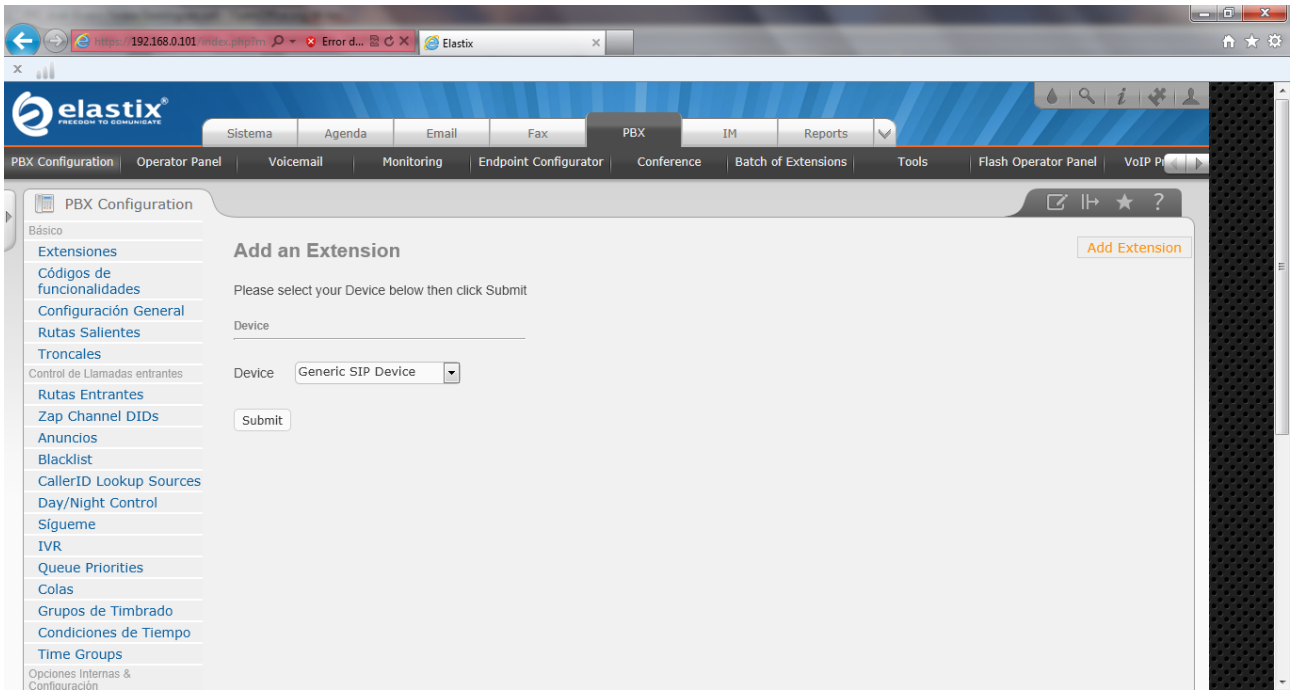
configurados mediante la interfaz web.

9.5. Configuración

Para acceder al interfaz de administración, se abre un un navegador y se teclea la dirección del servidor Elastix: https://192.168.0.101. Una vez introducidas las credenciales indicadas durante la instalación, sale el Panel de control, o Dashboard, de Elastix:



Esta pantalla, organizada en componentes similares a los portlets, sirve para ver de un vistazo el estado del sistema. Pulsando sobre PBX, accedemos a la configuración de Asterisk, en la que nos aparece en primer lugar la opción para añadir una extensión:



Para la parte de la configuración PBX, Elastix ha integrado freePBX. freePBX es un software de configuración y control de Asterisk, cuyo fin es facilitar mediante un interfaz gráfico (GUI), las labores habituales de mantenimiento. Genera los archivos de configuración de Asterisk, incluido el plan de marcado, en función de la configuración que el usuario introduce desde la Web. Dicha configuración se almacena en una base de datos MySql, y es convertida en archivos

de configuración de Asterisk mediante una aplicación llamada retrieve_conf. Cada vez que pulsamos en la barrita roja que dice "Apply Configuration Changes Here", se invoca al script retrieve_conf, que también puede ser llamado a través de la línea de comandos.

En primer lugar, se añade una extensión SIP, seleccionando "Generic SIP Device" y pulsando Submit. En la pantalla "Add SIP Extension", rellenamos los siguientes campos con estos valores:

- Device: Generic SIP Device
- User Extension: 3000
- Display Name: Extension 1
- secret: 3000 ; para un sistema en producción hay que elegir una contraseña más segura
- Language Code: es ; para que Asterisk nos de sus mensajes en español

Una vez enviados los cambios con Submit, es importante aplicarlos pulsando sobre el enlace "Apply Configuration Changes Here". En este caso, añadiremos dos extensiones más antes de aplicar la nueva configuración con la siguiente configuración:

- Device: Generic IAX2 Device
- User Extension: 3001
- Display Name: Extension 2
- secret: 3001
- Language Code: es

- Device: Generic SIP Device
- User Extension: 3002
- Display Name: Extension 3
- secret: 3002
- Language Code: es

Una vez aplicada la configuración, el fichero sip_additional.conf tiene el siguiente contenido:

```
[root@elastix ~]# more /etc/asterisk/sip_additional.conf
;------------------------------------------------------------------------------;
; Do NOT edit this file as it is auto-generated by FreePBX. All modifications to ;
; this file must be done via the web gui. There are alternative files to make    ;
; custom modifications, details at: http://freepbx.org/configuration_files       ;
;------------------------------------------------------------------------------;
;
```

```
[3000]
deny=0.0.0.0/0.0.0.0
secret=3000
dtmfmode=rfc2833
canreinvite=no
context=from-internal
host=dynamic
type=friend
nat=yes
port=5060
qualify=yes
callgroup=
pickupgroup=
dial=SIP/3000
mailbox=3000@device
permit=0.0.0.0/0.0.0.0
callerid=device <3000>
callcounter=yes
faxdetect=no

[3002]
deny=0.0.0.0/0.0.0.0
secret=3002
dtmfmode=rfc2833
canreinvite=no
context=from-internal
host=dynamic
type=friend
nat=yes
port=5060
qualify=yes
callgroup=
pickupgroup=
```

dial=SIP/3002

mailbox=3002@device

permit=0.0.0.0/0.0.0.0

callerid=device <3002>

callcounter=yes

faxdetect=no

Y el archivo iax_additional.conf:

[root@elastix ~]# more /etc/asterisk/iax_additional.conf

;--;

; Do NOT edit this file as it is auto-generated by FreePBX. All modifications to ;

; this file must be done via the web gui. There are alternative files to make ;

; custom modifications, details at: http://freepbx.org/configuration_files ;

;--;

;

[3001]

deny=0.0.0.0/0.0.0.0

secret=3001

transfer=no

context=from-internal

host=dynamic

type=friend

port=4569

qualify=yes

dial=IAX2/3001

mailbox=3001@device

permit=0.0.0.0/0.0.0.0

callerid=device <3001>

setvar=REALCALLERIDNUM=3001

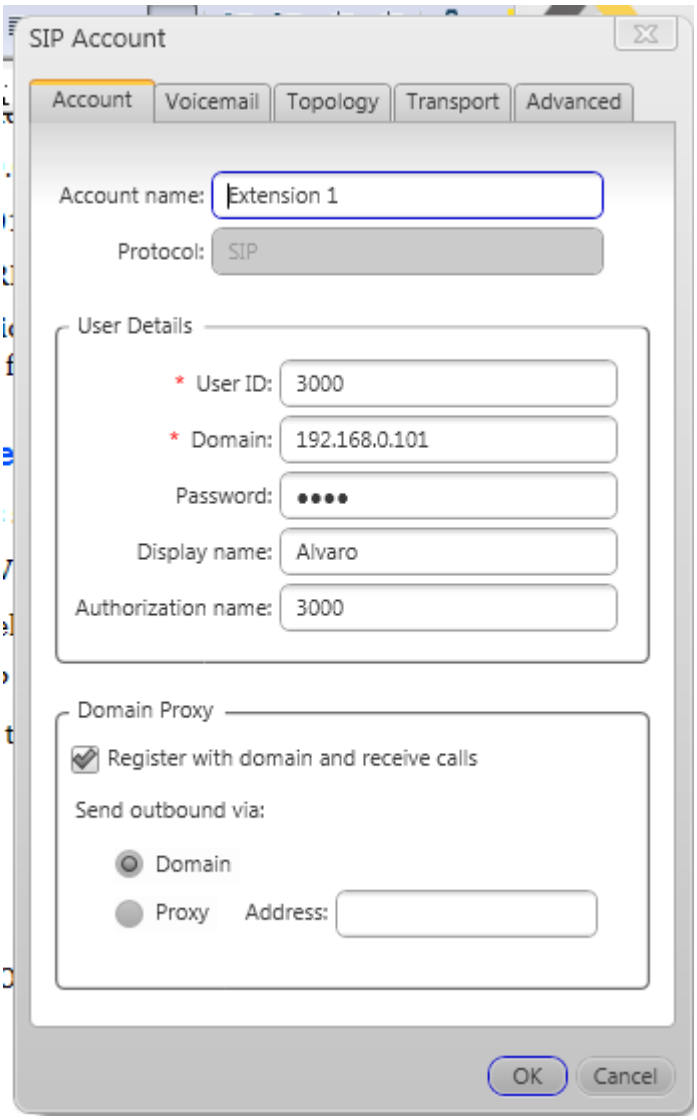
Con esta configuración, Asterisk ya tiene preparadas tres extensiones para hacer y recibir llamadas por ellas, a falta de terminales que las usen.

9.6. Instalación de softphones

El primer de los softphones instalados tiene las siguientes características:

- Plataforma: Windows
- Marca y modelo: CounterPath X-Lite 5.0
- Protocolo: SIP

La configuración del teléfono se hace en apartado Softphone -> Account Settings:



Una vez aplicados los cambios, en log de Asterisk (/var/log/asterisk/full) se ve la siguiente entrada:

[Dec 31 20:31:12] NOTICE[4144] chan_sip.c: Peer '3000' is now Reachable. (10ms / 2000ms)

Que indica que el teléfono se ha registrado bien.

El segundo softphone tiene las siguientes características:

- Plataforma: Android 2.3.5 en un Samsung GT-S5830

- Marca y modelo: Zoiper 1.10 for Android
- Protocolo: SIP

La configuración del teléfono se hace, dentro del programa, en apartado Config -> Accounts -> Add account:

Una vez rellenados los datos y aplicados los cambios, en log de Asterisk (/var/log/asterisk/full) se ve la siguiente entrada:

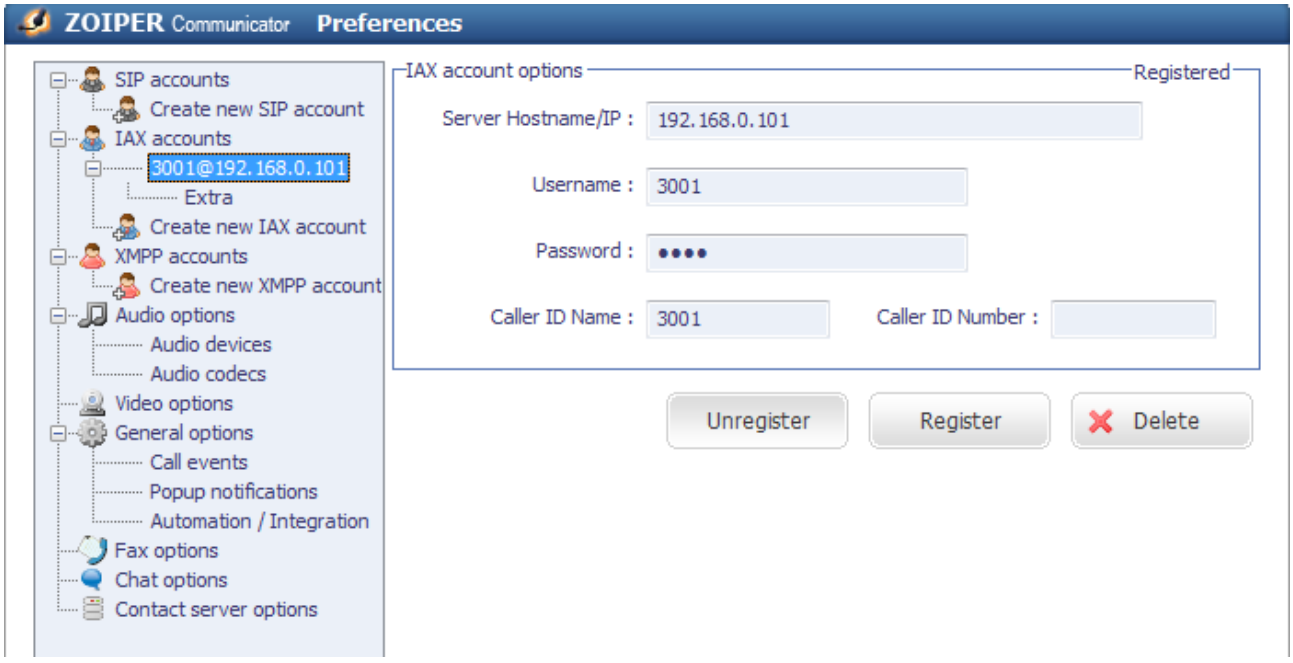
[Dec 31 20:58:50] VERBOSE[4144] chan_sip.c: -- Registered SIP '3002' at 192.168.0.102:5060

[Dec 31 20:58:50] NOTICE[4144] chan_sip.c: Peer '3002' is now Reachable. (52ms / 2000ms)

El tercer softphone tiene las siguientes características:

- Plataforma: Windows
- Marca y modelo: Zoiper Communicator 2.05
- Protocolo: IAX2

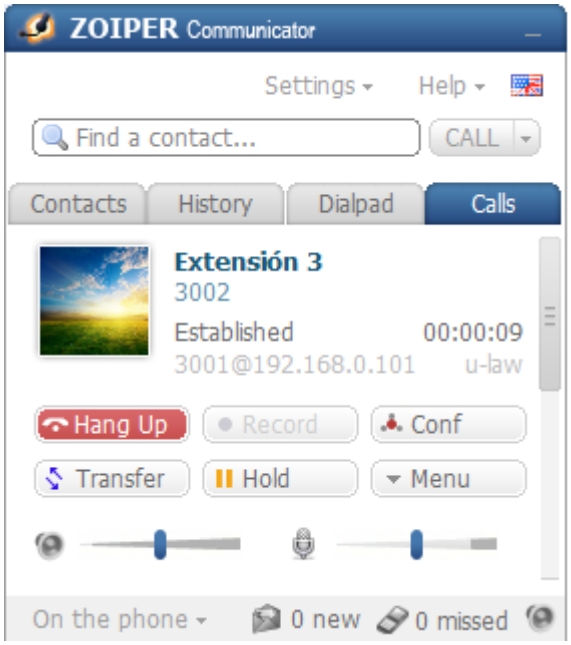
La configuración del teléfono se hace en Settings -> Preferences.



Una vez añadida la cuenta, en el log aparece la siguiente entrada que indica que se ha registrado bien el teléfono:

[Dec 31 21:04:29] VERBOSE[4131] chan_iax2.c: -- Registered IAX2 '3001' (AUTHENTICATED) at 192.168.0.100:4569

Para probar la configuración, se realiza una llamada desde la Extensión 2 (3001) a la Extensión 3 (3002). Una vez establecida la llamada, el interfaz de usuario de Extensión 2, que usa Zoiper en Windows, es:



En Asterisk, el log de la llamada es el siguiente:

[Jan 1 18:27:24] VERBOSE[4130] chan_iax2.c: -- Accepting AUTHENTICATED call from 192.168.0.100:

> requested format = gsm,

> requested prefs = (),

> actual format = ulaw,

> host prefs = (ulaw|alaw|gsm),

> priority = mine

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [3002@from-internal:1] Macro("IAX2/3001-891", "exten-vm,novm,3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:1] Macro("IAX2/3001-891", "user-callerid,") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:1] Set("IAX2/3001-891", "AMPUSER=3001") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:2] GotoIf("IAX2/3001-891", "0?report") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:3] ExecIf("IAX2/3001-891", "0?Set(REALCALLERIDNUM=3001)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:4] Set("IAX2/3001-891", "AMPUSER=3001") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:5] Set("IAX2/3001-891", "AMPUSERCIDNAME=Extension 2") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:6] GotoIf("IAX2/3001-891", "0?report") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:7] Set("IAX2/3001-891", "AMPUSERCID=3001") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:8] Set("IAX2/3001-891", "CALLERID(all)="Extension 2" <3001>") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:9] ExecIf("IAX2/3001-891", "1?Set(CHANNEL(language)=es)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:10] GotoIf("IAX2/3001-891", "0?continue") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:11] Set("IAX2/3001-891", "__TTL=64") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:12] GotoIf("IAX2/3001-891", "1?continue") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-user-callerid,s,19)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:19] Set("IAX2/3001-891", "CALLERID(number)=3001") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:20] Set("IAX2/3001-891", "CALLERID(name)=Extension 2") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-user-callerid:21] NoOp("IAX2/3001-891", "Using CallerID "Extension 2" <3001>") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:2] Set("IAX2/3001-891", "RingGroupMethod=none") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:3] Set("IAX2/3001-891", "VMBOX=novm") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:4] Set("IAX2/3001-891", "__EXTTOCALL=3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:5] Set("IAX2/3001-891", "CFUEXT=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:6] Set("IAX2/3001-891", "CFBEXT=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:7] Set("IAX2/3001-891", "RT="") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:8] Macro("IAX2/3001-891", "record-enable,3002,IN") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-record-enable:1] GotoIf("IAX2/3001-891", "1?check") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-record-enable,s,4)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-record-enable:4] ExecIf("IAX2/3001-891", "0?MacroExit()") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-record-enable:5] GotoIf("IAX2/3001-891", "0?Group:OUT") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-record-enable,s,15)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-record-enable:15] GotoIf("IAX2/3001-891", "1?IN") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-record-enable,s,20)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-record-enable:20] ExecIf("IAX2/3001-891", "1?MacroExit()") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-exten-vm:9] Macro("IAX2/3001-891", "dial-one,"",r,3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:1] Set("IAX2/3001-891", "DEXTEN=3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:2] Set("IAX2/3001-891", "DIALSTATUS_CW=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:3] GosubIf("IAX2/3001-891", "0?screen,1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:4] GosubIf("IAX2/3001-891", "0?cf,1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:5] GotoIf("IAX2/3001-891", "1?skip1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,8)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:8] GotoIf("IAX2/3001-891", "0?nodial") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:9] GotoIf("IAX2/3001-891", "0?continue") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:10] Set("IAX2/3001-891", "EXTHASCW=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:11] GotoIf("IAX2/3001-891", "1?next1:cwinusebusy") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,12)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:12] GotoIf("IAX2/3001-891", "0?docfu:skip3") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,16)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:16] GotoIf("IAX2/3001-891", "1?next2:continue") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,17)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:17] GotoIf("IAX2/3001-891", "1?continue") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,25)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:25] GotoIf("IAX2/3001-891", "0?nodial") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:26] GosubIf("IAX2/3001-891", "1?dstring,1:dlocal,1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:1] Set("IAX2/3001-891", "DSTRING=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:2] Set("IAX2/3001-891", "DEVICES=3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:3] ExecIf("IAX2/3001-891", "0?Return()") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:4] ExecIf("IAX2/3001-891", "0?Set(DEVICES=002)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:5] Set("IAX2/3001-891", "LOOPCNT=1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:6] Set("IAX2/3001-891", "ITER=1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:7] Set("IAX2/3001-891", "THISDIAL=SIP/3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:8] GosubIf("IAX2/3001-891", "1?zap2dahdi,1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:1] ExecIf("IAX2/3001-891", "0?Return()") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:2] Set("IAX2/3001-891", "NEWDIAL=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:3] Set("IAX2/3001-891", "LOOPCNT2=1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:4] Set("IAX2/3001-891", "ITER2=1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:5] Set("IAX2/3001-891", "THISPART2=SIP/3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:6] ExecIf("IAX2/3001-891", "0?Set(THISPART2=DAHDI/3002)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:7] Set("IAX2/3001-891", "NEWDIAL=SIP/3002&") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:8] Set("IAX2/3001-891", "ITER2=2") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:9] GotoIf("IAX2/3001-891", "0?begin2") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:10] Set("IAX2/3001-891", "THISDIAL=SIP/3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [zap2dahdi@macro-dial-one:11] Return("IAX2/3001-891", "") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:9] Set("IAX2/3001-891", "DSTRING=SIP/3002&") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:10] Set("IAX2/3001-891", "ITER=2") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:11] GotoIf("IAX2/3001-891", "0?begin") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:12] Set("IAX2/3001-891", "DSTRING=SIP/3002") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [dstring@macro-dial-one:13] Return("IAX2/3001-891", "") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:27] GotoIf("IAX2/3001-891", "0?nodial") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:28] GotoIf("IAX2/3001-891", "1?skiptrace") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Goto (macro-dial-one,s,30)

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:30] Set("IAX2/3001-891", "D_OPTIONS=r") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:31] ExecIf("IAX2/3001-891", "0?SIPAddHeader(Alert-Info:)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:32] ExecIf("IAX2/3001-891", "0?SIPAddHeader()") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:33] ExecIf("IAX2/3001-891", "0?Set(CHANNEL(musicclass)=)") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:34] GosubIf("IAX2/3001-891", "0?qwait,1") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:35] Set("IAX2/3001-891", "__CWIGNORE=") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:36] Set("IAX2/3001-891", "__KEEPCID=TRUE") in new stack

[Jan 1 18:27:24] VERBOSE[4688] pbx.c: -- Executing [s@macro-dial-one:37] Dial("IAX2/3001-891", "SIP/3002,"",r") in new stack

[Jan 1 18:27:24] VERBOSE[4688] netsock2.c: == Using SIP RTP TOS bits 184

[Jan 1 18:27:24] VERBOSE[4688] netsock2.c: == Using SIP RTP CoS mark 5

[Jan 1 18:27:24] VERBOSE[4688] app_dial.c: -- Called SIP/3002

[Jan 1 18:27:24] VERBOSE[4688] app_dial.c: -- SIP/3002-00000000 is ringing

[Jan 1 18:27:24] VERBOSE[4688] app_dial.c: -- SIP/3002-00000000 is ringing

[Jan 1 18:27:28] VERBOSE[4688] app_dial.c: -- SIP/3002-00000000 answered IAX2/3001-891

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [h@macro-dial-one:1] Macro("IAX2/3001-891", "hangupcall,") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:1] GotoIf("IAX2/3001-891", "1?endmixmoncheck") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,9)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:9] NoOp("IAX2/3001-891", "End of MIXMON check") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:10] GotoIf("IAX2/3001-891", "1?nomeetmemon") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,15)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:15] NoOp("IAX2/3001-891", "MEETME_RECORDINGFILE=") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:16] GotoIf("IAX2/3001-891", "1?noautomon") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,18)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:18] NoOp("IAX2/3001-891", "TOUCH_MONITOR_OUTPUT=") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:19] GotoIf("IAX2/3001-891", "1?noautomon2") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,25)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:25] NoOp("IAX2/3001-891", "MONITOR_FILENAME=") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:26] GotoIf("IAX2/3001-891", "1?skiprg") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,29)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:29] GotoIf("IAX2/3001-891", "1?skipblkvm") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,32)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:32] GotoIf("IAX2/3001-891", "1?theend") in new stack

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Goto (macro-hangupcall,s,34)

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: -- Executing [s@macro-hangupcall:34] Hangup("IAX2/3001-891", "") in new stack

[Jan 1 18:28:52] VERBOSE[4688] app_macro.c: == Spawn extension (macro-hangupcall, s, 34) exited non-zero on 'IAX2/3001-891' in macro 'hangupcall'

[Jan 1 18:28:52] VERBOSE[4688] features.c: == Spawn extension (macro-dial-one, h, 1) exited non-zero on 'IAX2/3001-891'

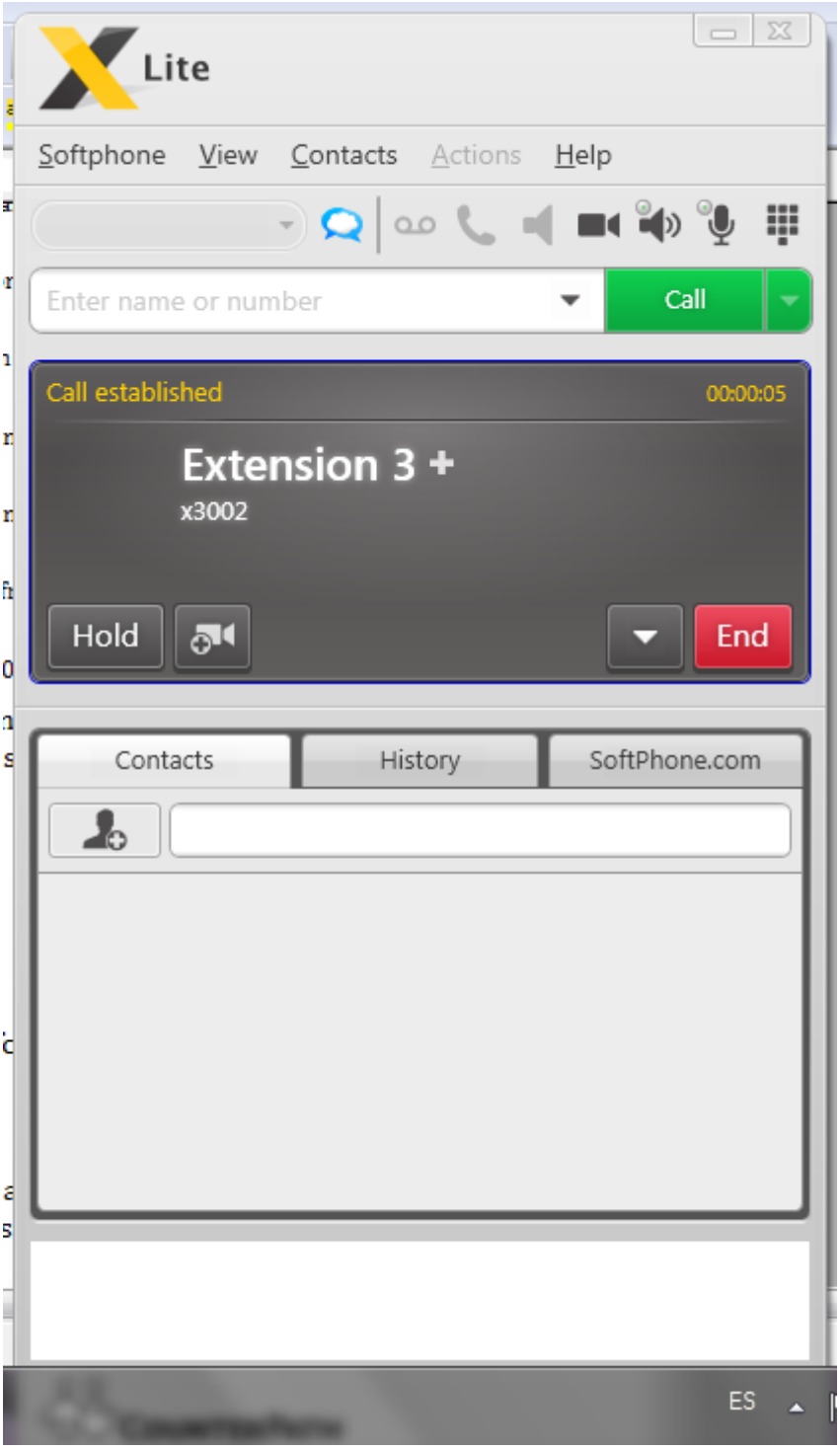
[Jan 1 18:28:52] VERBOSE[4688] app_macro.c: == Spawn extension (macro-dial-one, s, 37) exited non-zero on 'IAX2/3001-891' in macro 'dial-one'

[Jan 1 18:28:52] VERBOSE[4688] app_macro.c: == Spawn extension (macro-exten-vm, s, 9) exited non-zero on 'IAX2/3001-891' in macro 'exten-vm'

[Jan 1 18:28:52] VERBOSE[4688] pbx.c: == Spawn extension (from-internal, 3002, 1) exited non-zero on 'IAX2/3001-891'

[Jan 1 18:28:52] VERBOSE[4688] chan_iax2.c: -- Hungup 'IAX2/3001-891'

Para la siguiente prueba se hace una llamada desde Extensión 3 (3002) a Extensión 1 (3000). El interfaz de CounterPath X-Lite 5.0 en Windows con la llamada en curso es el siguiente:



El log de la llamada es:

[Jan 1 18:40:30] VERBOSE[4140] netsock2.c: == Using SIP RTP TOS bits 184

[Jan 1 18:40:30] VERBOSE[4140] netsock2.c: == Using SIP RTP CoS mark 5

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [3000@from-internal:1] Macro("SIP/3002-00000001", "exten-vm,novm,3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:1] Macro("SIP/3002-00000001", "user-callerid,") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:1] Set("SIP/3002-00000001", "AMPUSER=3002") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:2] GotoIf("SIP/3002-00000001", "0?report") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:3] ExecIf("SIP/3002-00000001", "1?Set(REALCALLERIDNUM=3002)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:4] Set("SIP/3002-00000001", "AMPUSER=3002") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:5] Set("SIP/3002-00000001", "AMPUSERCIDNAME=Extension 3") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:6] GotoIf("SIP/3002-00000001", "0?report") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:7] Set("SIP/3002-00000001", "AMPUSERCID=3002") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:8] Set("SIP/3002-00000001", "CALLERID(all)="Extension 3" <3002>") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:9] ExecIf("SIP/3002-00000001", "1?Set(CHANNEL(language)=es)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:10] GotoIf("SIP/3002-00000001", "0?continue") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:11] Set("SIP/3002-00000001", "__TTL=64") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:12] GotoIf("SIP/3002-00000001", "1?continue") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-user-callerid,s,19)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:19] Set("SIP/3002-00000001", "CALLERID(number)=3002") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:20] Set("SIP/3002-00000001", "CALLERID(name)=Extension 3") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-user-callerid:21] NoOp("SIP/3002-00000001", "Using CallerID "Extension 3" <3002>") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:2] Set("SIP/3002-00000001", "RingGroupMethod=none") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:3] Set("SIP/3002-00000001", "VMBOX=novm") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:4] Set("SIP/3002-00000001", "__EXTTOCALL=3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:5] Set("SIP/3002-00000001", "CFUEXT=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:6] Set("SIP/3002-00000001", "CFBEXT=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:7] Set("SIP/3002-00000001",

"RT="""") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:8] Macro("SIP/3002-00000001", "record-enable,3000,IN") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-record-enable:1] GotoIf("SIP/3002-00000001", "1?check") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-record-enable,s,4)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-record-enable:4] ExecIf("SIP/3002-00000001", "0?MacroExit()") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-record-enable:5] GotoIf("SIP/3002-00000001", "0?Group:OUT") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-record-enable,s,15)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-record-enable:15] GotoIf("SIP/3002-00000001", "1?IN") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-record-enable,s,20)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-record-enable:20] ExecIf("SIP/3002-00000001", "1?MacroExit()") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-exten-vm:9] Macro("SIP/3002-00000001", "dial-one,"",r,3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:1] Set("SIP/3002-00000001", "DEXTEN=3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:2] Set("SIP/3002-00000001", "DIALSTATUS_CW=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:3] GosubIf("SIP/3002-00000001", "0?screen,1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:4] GosubIf("SIP/3002-00000001", "0?cf,1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:5] GotoIf("SIP/3002-00000001", "1?skip1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,8)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:8] GotoIf("SIP/3002-00000001", "0?nodial") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:9] GotoIf("SIP/3002-00000001", "0?continue") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:10] Set("SIP/3002-00000001", "EXTHASCW=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:11] GotoIf("SIP/3002-00000001", "1?next1:cwinusebusy") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,12)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:12] GotoIf("SIP/3002-00000001", "0?docfu:skip3") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,16)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:16] GotoIf("SIP/3002-00000001", "1?next2:continue") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,17)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:17] GotoIf("SIP/3002-00000001", "1?continue") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,25)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:25] GotoIf("SIP/3002-00000001", "0?nodial") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:26] GosubIf("SIP/3002-00000001", "1?dstring,1:dlocal,1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:1] Set("SIP/3002-00000001", "DSTRING=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:2] Set("SIP/3002-00000001", "DEVICES=3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:3] ExecIf("SIP/3002-00000001", "0?Return()") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:4] ExecIf("SIP/3002-00000001", "0?Set(DEVICES=000)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:5] Set("SIP/3002-00000001", "LOOPCNT=1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:6] Set("SIP/3002-00000001", "ITER=1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:7] Set("SIP/3002-00000001", "THISDIAL=SIP/3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:8] GosubIf("SIP/3002-00000001", "1?zap2dahdi,1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:1] ExecIf("SIP/3002-00000001", "0?Return()") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:2] Set("SIP/3002-00000001", "NEWDIAL=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:3] Set("SIP/3002-00000001", "LOOPCNT2=1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:4] Set("SIP/3002-00000001", "ITER2=1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:5] Set("SIP/3002-00000001", "THISPART2=SIP/3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:6] ExecIf("SIP/3002-00000001", "0?Set(THISPART2=DAHDI/3000)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:7] Set("SIP/3002-00000001", "NEWDIAL=SIP/3000&") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:8] Set("SIP/3002-00000001", "ITER2=2") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:9] GotoIf("SIP/3002-00000001", "0?begin2") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:10] Set("SIP/3002-00000001", "THISDIAL=SIP/3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [zap2dahdi@macro-dial-one:11] Return("SIP/3002-00000001", "") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:9] Set("SIP/3002-00000001", "DSTRING=SIP/3000&") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:10] Set("SIP/3002-00000001", "ITER=2") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:11] GotoIf("SIP/3002-00000001", "0?begin") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:12] Set("SIP/3002-00000001", "DSTRING=SIP/3000") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [dstring@macro-dial-one:13] Return("SIP/3002-00000001", "") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:27] GotoIf("SIP/3002-00000001", "0?nodial") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:28] GotoIf("SIP/3002-00000001", "1?skiptrace") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Goto (macro-dial-one,s,30)

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:30] Set("SIP/3002-00000001", "D_OPTIONS=r") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:31] ExecIf("SIP/3002-00000001", "0?SIPAddHeader(Alert-Info:)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:32] ExecIf("SIP/3002-00000001", "0?SIPAddHeader()") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:33] ExecIf("SIP/3002-00000001", "0?Set(CHANNEL(musicclass)=)") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:34] GosubIf("SIP/3002-00000001", "0?qwait,1") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:35] Set("SIP/3002-00000001", "__CWIGNORE=") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:36] Set("SIP/3002-00000001", "__KEEPCID=TRUE") in new stack

[Jan 1 18:40:30] VERBOSE[4709] pbx.c: -- Executing [s@macro-dial-one:37] Dial("SIP/3002-00000001", "SIP/3000,"",r") in new stack

[Jan 1 18:40:30] VERBOSE[4709] netsock2.c: == Using SIP RTP TOS bits 184

[Jan 1 18:40:30] VERBOSE[4709] netsock2.c: == Using SIP RTP CoS mark 5

[Jan 1 18:40:30] VERBOSE[4709] app_dial.c: -- Called SIP/3000

[Jan 1 18:40:31] VERBOSE[4709] app_dial.c: -- SIP/3000-00000002 is ringing

[Jan 1 18:40:37] VERBOSE[4709] app_dial.c: -- SIP/3000-00000002 answered SIP/3002-00000001

[Jan 1 18:40:37] VERBOSE[4709] rtp_engine.c: -- Locally bridging SIP/3002-00000001 and SIP/3000-00000002

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [h@macro-dial-one:1] Macro("SIP/3002-00000001", "hangupcall,") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:1] GotoIf("SIP/3002-00000001",

"1?endmixmoncheck") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,9)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:9] NoOp("SIP/3002-00000001", "End of MIXMON check") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:10] GotoIf("SIP/3002-00000001", "1?nomeetmemon") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,15)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:15] NoOp("SIP/3002-00000001", "MEETME_RECORDINGFILE=") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:16] GotoIf("SIP/3002-00000001", "1?noautomon") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,18)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:18] NoOp("SIP/3002-00000001", "TOUCH_MONITOR_OUTPUT=") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:19] GotoIf("SIP/3002-00000001", "1?noautomon2") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,25)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:25] NoOp("SIP/3002-00000001", "MONITOR_FILENAME=") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:26] GotoIf("SIP/3002-00000001", "1?skiprg") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,29)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:29] GotoIf("SIP/3002-00000001", "1?skipblkvm") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,32)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:32] GotoIf("SIP/3002-00000001", "1?theend") in new stack

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Goto (macro-hangupcall,s,34)

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: -- Executing [s@macro-hangupcall:34] Hangup("SIP/3002-00000001", "") in new stack

[Jan 1 18:42:32] VERBOSE[4709] app_macro.c: == Spawn extension (macro-hangupcall, s, 34) exited non-zero on 'SIP/3002-00000001' in macro 'hangupcall'

[Jan 1 18:42:32] VERBOSE[4709] features.c: == Spawn extension (macro-dial-one, h, 1) exited non-zero on 'SIP/3002-00000001'

[Jan 1 18:42:32] VERBOSE[4709] app_macro.c: == Spawn extension (macro-dial-one, s, 37) exited non-zero on 'SIP/3002-00000001' in macro 'dial-one'

[Jan 1 18:42:32] VERBOSE[4709] app_macro.c: == Spawn extension (macro-exten-vm, s, 9) exited non-zero on 'SIP/3002-00000001' in macro 'exten-vm'

[Jan 1 18:42:32] VERBOSE[4709] pbx.c: == Spawn extension (from-internal, 3000, 1) exited non-zero on 'SIP/3002-00000001'

Se hace otra prueba más desde Extensión 1 (3000) a Extensión 2 (3001), en la que se devolverá el tono comunicando. El log de la llamada es el siguiente:

[Jan 1 18:51:51] VERBOSE[4140] netsock2.c: == Using SIP RTP TOS bits 184

[Jan 1 18:51:51] VERBOSE[4140] netsock2.c: == Using SIP RTP CoS mark 5

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [3001@from-internal:1] Macro("SIP/3000-00000009", "exten-vm,novm,3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:1] Macro("SIP/3000-00000009", "user-callerid,") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:1] Set("SIP/3000-00000009", "AMPUSER=3000") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:2] GotoIf("SIP/3000-00000009", "0?report") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:3] ExecIf("SIP/3000-00000009", "1?Set(REALCALLERIDNUM=3000)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:4] Set("SIP/3000-00000009", "AMPUSER=3000") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:5] Set("SIP/3000-00000009", "AMPUSERCIDNAME=Extension 1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:6] GotoIf("SIP/3000-00000009", "0?report") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:7] Set("SIP/3000-00000009", "AMPUSERCID=3000") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:8] Set("SIP/3000-00000009", "CALLERID(all)="Extension 1" <3000>") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:9] ExecIf("SIP/3000-00000009", "1?Set(CHANNEL(language)=es)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:10] GotoIf("SIP/3000-00000009", "0?continue") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:11] Set("SIP/3000-00000009", "__TTL=64") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:12] GotoIf("SIP/3000-00000009", "1?continue") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-user-callerid,s,19)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:19] Set("SIP/3000-00000009", "CALLERID(number)=3000") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:20] Set("SIP/3000-00000009", "CALLERID(name)=Extension 1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-user-callerid:21] NoOp("SIP/3000-00000009", "Using CallerID "Extension 1" <3000>") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:2] Set("SIP/3000-00000009", "RingGroupMethod=none") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:3] Set("SIP/3000-00000009", "VMBOX=novm") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:4] Set("SIP/3000-00000009", "__EXTTOCALL=3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:5] Set("SIP/3000-00000009",

"CFUEXT=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:6] Set("SIP/3000-00000009", "CFBEXT=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:7] Set("SIP/3000-00000009", "RT="") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:8] Macro("SIP/3000-00000009", "record-enable,3001,IN") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-record-enable:1] GotoIf("SIP/3000-00000009", "1?check") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-record-enable,s,4)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-record-enable:4] ExecIf("SIP/3000-00000009", "0?MacroExit()") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-record-enable:5] GotoIf("SIP/3000-00000009", "0?Group:OUT") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-record-enable,s,15)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-record-enable:15] GotoIf("SIP/3000-00000009", "1?IN") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-record-enable,s,20)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-record-enable:20] ExecIf("SIP/3000-00000009", "1?MacroExit()") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:9] Macro("SIP/3000-00000009", "dial-one,"",r,3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:1] Set("SIP/3000-00000009", "DEXTEN=3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:2] Set("SIP/3000-00000009", "DIALSTATUS_CW=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:3] GosubIf("SIP/3000-00000009", "0?screen,1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:4] GosubIf("SIP/3000-00000009", "0?cf,1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:5] GotoIf("SIP/3000-00000009", "1?skip1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,8)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:8] GotoIf("SIP/3000-00000009", "0?nodial") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:9] GotoIf("SIP/3000-00000009", "0?continue") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:10] Set("SIP/3000-00000009", "EXTHASCW=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:11] GotoIf("SIP/3000-00000009", "1?next1:cwinusebusy") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,12)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:12] GotoIf("SIP/3000-00000009", "0?

docfu:skip3") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,16)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:16] GotoIf("SIP/3000-00000009", "1?next2:continue") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,17)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:17] GotoIf("SIP/3000-00000009", "1?continue") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,25)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:25] GotoIf("SIP/3000-00000009", "0?nodial") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:26] GosubIf("SIP/3000-00000009", "1?dstring,1:dlocal,1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:1] Set("SIP/3000-00000009", "DSTRING=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:2] Set("SIP/3000-00000009", "DEVICES=3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:3] ExecIf("SIP/3000-00000009", "0?Return()") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:4] ExecIf("SIP/3000-00000009", "0?Set(DEVICES=001)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:5] Set("SIP/3000-00000009", "LOOPCNT=1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:6] Set("SIP/3000-00000009", "ITER=1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:7] Set("SIP/3000-00000009", "THISDIAL=IAX2/3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:8] GosubIf("SIP/3000-00000009", "1?zap2dahdi,1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:1] ExecIf("SIP/3000-00000009", "0?Return()") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:2] Set("SIP/3000-00000009", "NEWDIAL=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:3] Set("SIP/3000-00000009", "LOOPCNT2=1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:4] Set("SIP/3000-00000009", "ITER2=1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:5] Set("SIP/3000-00000009", "THISPART2=IAX2/3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:6] ExecIf("SIP/3000-00000009", "0?Set(THISPART2=DAHDI2/3001)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:7] Set("SIP/3000-00000009", "NEWDIAL=IAX2/3001&") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:8] Set("SIP/3000-00000009", "ITER2=2") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:9] GotoIf("SIP/3000-00000009", "0?begin2") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:10] Set("SIP/3000-00000009", "THISDIAL=IAX2/3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [zap2dahdi@macro-dial-one:11] Return("SIP/3000-00000009", "") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:9] Set("SIP/3000-00000009", "DSTRING=IAX2/3001&") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:10] Set("SIP/3000-00000009", "ITER=2") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:11] GotoIf("SIP/3000-00000009", "0?begin") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:12] Set("SIP/3000-00000009", "DSTRING=IAX2/3001") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [dstring@macro-dial-one:13] Return("SIP/3000-00000009", "") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:27] GotoIf("SIP/3000-00000009", "0?nodial") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:28] GotoIf("SIP/3000-00000009", "1?skiptrace") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Goto (macro-dial-one,s,30)

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:30] Set("SIP/3000-00000009", "D_OPTIONS=r") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:31] ExecIf("SIP/3000-00000009", "0?SIPAddHeader(Alert-Info:)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:32] ExecIf("SIP/3000-00000009", "0?SIPAddHeader()") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:33] ExecIf("SIP/3000-00000009", "0?Set(CHANNEL(musicclass)=)") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:34] GosubIf("SIP/3000-00000009", "0?qwait,1") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:35] Set("SIP/3000-00000009", "__CWIGNORE=") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:36] Set("SIP/3000-00000009", "__KEEPCID=TRUE") in new stack

[Jan 1 18:51:51] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:37] Dial("SIP/3000-00000009", "IAX2/3001,"",r") in new stack

[Jan 1 18:51:51] VERBOSE[4726] app_dial.c: -- Called IAX2/3001

[Jan 1 18:51:51] VERBOSE[4130] chan_iax2.c: -- Call accepted by 192.168.0.100 (format ulaw)

[Jan 1 18:51:51] VERBOSE[4130] chan_iax2.c: -- Format for call is ulaw

[Jan 1 18:51:51] VERBOSE[4726] app_dial.c: -- IAX2/3001-6554 is ringing

[Jan 1 18:51:59] WARNING[4134] chan_iax2.c: Call rejected by 192.168.0.100: <Unknown>

[Jan 1 18:51:59] VERBOSE[4726] chan_iax2.c: -- Hungup 'IAX2/3001-6554'

[Jan 1 18:51:59] VERBOSE[4726] app_dial.c: == Everyone is busy/congested at this time (1:1/0/0)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:38] ExecIf("SIP/3000-00000009", "0?Set(DIALSTATUS=)") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:39] GosubIf("SIP/3000-00000009", "0?s-BUSY,1") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-dial-one:40] MacroExit("SIP/3000-00000009", "") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:10] GotoIf("SIP/3000-00000009", "0?exit") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:11] Set("SIP/3000-00000009", "SV_DIALSTATUS=BUSY") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:12] GosubIf("SIP/3000-00000009", "0?docfu,1") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:13] GosubIf("SIP/3000-00000009", "0?docfb,1") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:14] Set("SIP/3000-00000009", "DIALSTATUS=BUSY") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:15] NoOp("SIP/3000-00000009", "Voicemail is 'novm'") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-exten-vm:16] GotoIf("SIP/3000-00000009", "1?s-BUSY,1") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-exten-vm,s-BUSY,1)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s-BUSY@macro-exten-vm:1] NoOp("SIP/3000-00000009", "Extension is reporting BUSY and not passing to Voicemail") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s-BUSY@macro-exten-vm:2] GotoIf("SIP/3000-00000009", "0?exit,1") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s-BUSY@macro-exten-vm:3] PlayTones("SIP/3000-00000009", "busy") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s-BUSY@macro-exten-vm:4] Busy("SIP/3000-00000009", "20") in new stack

[Jan 1 18:51:59] VERBOSE[4726] app_macro.c: == Spawn extension (macro-exten-vm, s-BUSY, 4) exited non-zero on 'SIP/3000-00000009' in macro 'exten-vm'

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: == Spawn extension (from-internal, 3001, 1) exited non-zero on 'SIP/3000-00000009'

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [h@from-internal:1] Macro("SIP/3000-00000009", "hangupcall") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:1] GotoIf("SIP/3000-00000009", "1?endmixmoncheck") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,9)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:9] NoOp("SIP/3000-00000009", "End of MIXMON check") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:10] GotoIf("SIP/3000-00000009", "1?nomeetmemon") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,15)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:15] NoOp("SIP/3000-00000009", "MEETME_RECORDINGFILE=") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:16] GotoIf("SIP/3000-00000009", "1?noautomon") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,18)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:18] NoOp("SIP/3000-00000009", "TOUCH_MONITOR_OUTPUT=") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:19] GotoIf("SIP/3000-00000009", "1?noautomon2") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,25)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:25] NoOp("SIP/3000-00000009", "MONITOR_FILENAME=") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:26] GotoIf("SIP/3000-00000009", "1?skiprg") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,29)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:29] GotoIf("SIP/3000-00000009", "1?skipblkvm") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,32)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:32] GotoIf("SIP/3000-00000009", "1?theend") in new stack

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Goto (macro-hangupcall,s,34)

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: -- Executing [s@macro-hangupcall:34] Hangup("SIP/3000-00000009", "") in new stack

[Jan 1 18:51:59] VERBOSE[4726] app_macro.c: == Spawn extension (macro-hangupcall, s, 34) exited non-zero on 'SIP/3000-00000009' in macro 'hangupcall'

[Jan 1 18:51:59] VERBOSE[4726] pbx.c: == Spawn extension (from-internal, h, 1) exited non-zero on 'SIP/3000-00000009'

9.7. Calidad de voz

Los problemas más comunes en la telefonía, junto con la forma de abordarlos en Asterisk, son:

9.7.1. Eco

Consiste en que parte de la señal de ida se refleja en la de vuelta, lo que resulta bastante molesto. La causa más común en las líneas analógicas suele ser por un problema en el convertidor híbrido.

Otra causa consiste en que la impedancia de la línea telefónica varíe mucho, ya que bastantes de las tarjetas telefónicas para Asterisk no tienen buenos mecanismos de ajuste dinámico de la impedancia de la línea respecto de la tarjeta.

Asterisk dispone de una utilidad llamada fxotune que trata de ajustar la impedancia lo mejor posible, que debe ejecutarse con Asterisk bajado. El comando envía una señal por la línea y escucha el retorno, repitiendo el proceso hasta encontrar el mejor valor de ganacia, que escribe en /etc/fxotune.conf.

Otra causa del eco es cuando se retroalimenta el sonido desde el micrófono al auricular. En este caso, hay que solucionar el problema en el diseño del teléfono.

9.7.2. Volumen bajo

El volumen de recepción o transmisión pueden verse atenuados por las redes telefónicas de baja calidad, lo que afecta obviamente a la calidad de voz.

Esto se puede atenuar configurando el archivo de Asterisk zapata.conf, en el que se puede amplificar ambos volúmenes, el de recepción mediante el parámetro rxgain, y el de transmisión mediante txgain.

9.7.3. Demora

La demora es el tiempo que tarda la voz en llegar a su destino. Si es menor de 200 ms pasa desapercibida, pero si superan los 500 ms provocan problemas en la conversación, haciendo que se pisen los interlocutores.

Desafortunadamente, la demora no es posible mejorarla a nivel de servidor, ya que es un problema de la red de comunicaciones, por lo que habrá que analizar dicha red para ver donde se puede mejorar.

9.7.4. Distorsión de la voz

Este problema se percibe por parte de los usuarios como si el sonido estuviese robotizado. Por lo general, este efecto esta causado por usar un codec demasiado ahorrador en ancho de banda, como por ejemplo gsm. Si a la pérdida de información del proceso de codificación se le unen problemas con el ancho de banda, la distorsión será peor.

La solución consiste en cambiar de codec, teniendo en cuenta que el consumo de ancho de banda puede ser mayor si el codec lo requiere.

9.7.5. Comunicación entrecortada

La comunicación entrecortada tiene relación con la pérdida de paquetes, cuya causa suele ser una latencia elevada, un ancho de banda limitado o un jitter elevado en la red, y en cualquier caso problema de la red, no del servidor.

Normalmente, una latencia de 150 ms y una conexión de red no saturada no producirán comunicación entrecortada.

9.7.6. Jitter

El jitter se define como la variación entre el supuesto tiempo de llegada de un paquete, y su llegada real. No sólo es importante que el jitter sea bajo, sino también que no sea variable, ya que si lo es significa más probabilidad de que los paquetes lleguen desordenados. Al ser comunicaciones en tiempo real, el efecto en la conversación se nota enseguida, en forma de conversación entrecortada.

Asterisk tiene implementados buffers de jitter, vistos en el apartado 8.5, siendo configurables

en el archivo sip.conf y iax.conf, según el protocolo.

En el sip.conf:
jbenable=yes
jbmaxsize=200 ; tamaño del buffer

En el iax.conf:
jitterbuffer=yes
maxjitterbuffer=200 ; tamaño del buffer

9.8. Diagnóstico de problemas

El comienzo para diagnosticar cualquier problema es el log de Asterisk. Dicho log se configura en el archivo /etc/asterisk/logger.conf, donde se indican los destinos donde se va a enviar el log. Para cada uno de los destinos se puede configurar un determinado nivel de log. La sintaxis en logger.conf sería:

nombre_archivo => niveles separados por comas
donde nivel puede ser debug, notice, warning, error y verbose, según el nivel de detalle deseado.
Con Elastix, el archivo logger.conf contiene la línea:
full => notice, warning, error, debug, verbose
que configura un archivo llamado full en /var/log/asterisk. Cada línea de este archivo tiene el formato:
[HORA FECHA] NIVEL[PID] NOMBRE_ARCHIVO: TEXTO
Siendo el campo texto la información que, dependiendo del tipo de aviso que sea, tendrá un valor u otro. Pueden ser mensajes descriptivos con un error o un aviso, o información sobre algún proceso, por ejemplo el progreso de una llamada a través de un plan de marcado.

9.9. Seguridad

A continuación se exponen una serie de medidas muy básicas destinadas a mejorar la seguridad de una red VoIP.

- **Seguridad física:** Aunque pueda parecer algo muy obvio, el primer punto a tener en cuenta en la seguridad de un equipo es su seguridad física, de manera que nadie pueda, accidental o intencionadamente, dañar el servidor. Es importante al menos citar este punto, ya que no hace falta ser un experto para dañar un equipo. Para conseguir la seguridad física, se debe aislar el acceso físico al servidor Elastix, limitando el acceso a un grupo de personas mediante un sistema de autenticación.
- **Cambiar las claves por defecto en Elastix:** Aunque sea una medida de seguridad muy básica, no por ello hay que dejar de recordarla. En caso de no hacerlo, cualquiera con acceso a nuestro equipo puede introducirse en los programas y hacer lo que quiera.
- **Parar los servicios que no vamos a usar:** si se detienen todos los servicios que no se piensan usar, se reduce el número potencial de vulnerabilidades que pueden ser explotadas, además de liberar recursos innecesariamente ocupados.
- **Firewalls:** en todos los Linux viene el software iptables, que nos sirve para asegurar que el tráfico que recibe nuestro servidor sea el que, como administradores, tengamos

previsto.

- **IDS e IPS:** este tipo de programas va más allá que los cortafuegos, ya que miran dentro del paquete buscando patrones de ataques conocidos. Un IDS enviaría una notificación en caso de encontrar algo, mientras que un IPS podría lanzar alguna medida, como activar una regla de firewall.
- **Actualizaciones:** es importante tener siempre lo más actualizado posible el software que compone la distribución Elastix, debido a la corrección de errores que se produce en cada versión, muchos de los cuales arreglan problemas que comprometen la seguridad del equipo. También es importante estar pendiente de los avisos de seguridad que publica Asterisk en http://www.asterisk.org/security.

9.9.1. Ataques y vulnerabilidades IP

9.9.1.1. Denegación de servicio

Un ataque de Denegación de Servicio, o Denial of Service (DoS), es un ataque a una computadora o grupo de ellas, que están en red, provocando un consumo de alguno de los recursos excesivo, de modo que el sistema atacado no pueda cumplir su función. Los recursos atacados pueden ser, por ejemplo, el ancho de banda o los recursos computacionales. El método de ataque consiste en hacer una gran cantidad de peticiones al servidor, de manera que se sobrecargue y no sea capaz de seguir funcionando normalmente.

Una ampliación del DoS es el DDos, o DoS distribuido, que consiste en realizar el mismo ataque que en el DoS pero repetido desde múltiples clientes, pudiendo ser que estén coordinados para que el ataque llegue a ser masivo, siendo así más efectivo. A veces puede haber ataques de miles o decenas de miles de atacantes.

9.9.1.2. Ataques de SIP flooding

Es un tipo de ataque DoS que consiste en saturar el servidor Asterisk con paquetes INVITE. Estos paquetes sirven para iniciar una llamada, por lo que el servidor, cuando recibe uno de estos paquetes, reserva unos recursos y se queda esperando respuesta. Si se repite este proceso de envío de falsos paquetes INVITE, puede llegar un momento en que se produzca una denegación de servicio por saturación.

9.9.1.3. Sniffing

El sniffing consiste en recibir todos los paquetes que circulan por un switch. Por lo general, los switches tienen un modo monitorización que se puede activar para hacer que todo el tráfico que pasa por ellos se desvíe por un determinado puerto. En este puerto se utilizará un aplicación de esnifado, que básicamente lo que hace es filtrar el tráfico recibido buscando las características deseadas.

En caso de una red con tráfico SIP, es probable que se busque en los paquetes capturados información sobre, por ejemplo, Caller Ids, o incluso capturar el tráfico RTP y reproducir la llamada, ya que el tráfico RTP no se transmite encriptado.

La aplicación Wireshark es uno de los programas de esnifado más populares, e incorpora un plug-in para llamadas VoIP que graba la voz de los paquetes RTP.

Los problemas que pueda provocar el sniffing se soluciona usando encriptación de VoIP, por ejemplo usando el protocolo SRTP, soportado por Asterisk, o mediante VPN. En la actualidad ya existen teléfonos que pueden actuar como clientes VPN, o en caso de ser softphones, puede ser el mismo PC el que actúe como cliente.

En caso de usar el protocolo IAX, la encriptación consiste simplemente en añadir en la configuración IAX la línea "encryption=aes128".

10. Conclusiones

La telefonía es uno de los inventos que más impacto han tenido desde su aparición, hace menos de ciento cincuenta años. La velocidad con la que se ha expandido deja clara la capacidad que ha tenido para transformar la comunicación entre las personas. La inmediatez que proporciona a la hora de hablar con alguien, incluso estando en otra parte del mundo, ha cambiado para siempre la forma de relacionarse que tiene el ser humano, ya sea a nivel personal, como profesional, diplomático, institucional, informativo...

Además, sus redes han sido utilizadas para las comunicaciones entre ordenadores, pudiéndose decir también que dichas comunicaciones remotas han transformado, a la larga, la forma en que se usaban los sistemas de información.

Aunque los circuitos analógicos han iniciado todo este proceso, y lo han llevado muy lejos, hace ya cierto tiempo que los circuitos digitales habían ido tomando el relevo. Aunque en la "última milla" los circuitos analógicos sigan siendo mayoría, se puede decir que los circuitos digitales dominan en los clientes corporativos. En este sentido, han añadido ventajas como la multiplexión en el tiempo, que proporcionan un número mayor de líneas en una sola conexión. Además, al ser tecnología digital, aumenta la fiabilidad en la transmisión, permite técnicas de compresión, etc.

Estas dos tecnologías, analógicas y digitales, se habían desarrollado y habían crecido en redes de conmutación de circuitos, en las que como paso previo a la comunicación se debe establecer, por parte de los switches telefónicos, un circuito entre emisor y receptor, hasta que la comunicación termine.

Todo este escenario cambio por la telefonía IP y las redes de conmutación de paquetes. Toda la infraestructura y el conocimiento originado por el desarrollo de las redes TCP/IP, ya sea Internet o las redes locales, fue adaptada de manera natural a una nueva aplicación, la voz sobre IP. Se desarrollaron nuevos protocolos y estándares, como H.323, SIP y RTP, que intentasen compensar las posibles carencias de las redes de conmutación de paquetes, del protocolo IP, así como satisfacer las características de tiempo real de VoIP. Se han fabricado nuevos teléfonos que entiendan estos protocolos, y se ha desarrollado una nueva infraestructura para soportar estos nuevos sistemas de telefonía, así como la interconexión entre ellos.

Una vez llegados a este punto, VoIP está desplazando cada vez más a los sistemas de telefonía tradicionales. La consideración de que un terminal no es más que un dispositivo IP más, y la ubicuidad de este protocolo, hacen fácil la idea de que la extensión "siga" al usuario, ya sea en

un teléfono, un móvil, un ordenador, un televisor, o cualquier otro dispositivo que sea capaz de conectarse a una red IP. Además, el hecho de que esté en el mundo IP abre la puerta a cuantas aplicaciones seamos capaces de imaginar, ya que, en cierto modo, la telefonía abandona su habitat natural para meterse en el de los sistemas de información.

De hecho, una de las perjudicadas de esta revolución, a medio plazo, serán las compañías de telecomunicaciones, ya que no bastará con proveer de un teléfono y cobrar por las llamadas y los minutos, sino que deberán buscar algún valor añadido para hacer clientes, ya que la VoIP llegará a ser considerada como una "commodity".

Como no todo pueden ser ventajas, por experiencia, la implantación de VoIP en una organización algo grande supone un coste que hay que plantearse bien, ya que si no somos capaces de aportar ventajas claras de cara al negocio o al ahorro, podemos tener problemas para justificarla. Por ejemplo, el coste de la infraestructura de red necesaria para desplegar VoIP hay que tenerlo en cuenta, además del coste de gestión de tener, de repente, el doble de switches, que tienen que dar servicio a una aplicación en tiempo real. O por ejemplo, el coste de adquisición de terminales y el posible aprovechamiento de los antiguos.

En escenarios así, Asterisk puede ayudar muchísimo debido a su flexibilidad, ya que entiende tanto protocolos VoIP, como teléfonos y conexiones analógicas y digitales, ofreciéndonos una forma rápida de implantar nuestra propia PBX. Si optamos por una distribución como Elastix, tendremos además opción a obtener soporte comercial.