Universidad Nacional de La Pampa
Facultad de Ingeniería

Tesina para la Carrera

INGENIERIA EN SISTEMAS

Título:

IMPLEMENTACIÓN DE UNA CENTRAL TELEFÓNICA VOIP DE ALTA DISPONIBILIDAD EN UN AMBIENTE DISTRIBUIDO Y DE CALIDAD

Tesistas:

Fredes, Marcos A.

Mora, C. Nahuel

Mora, Hernán A.

Director:

Crespo, Aldo Abel

General Pico, La Pampa, 2012

Agradecimientos

Este trabajo de tesis, fruto de años de estudio y esfuerzo, pone punto final a esta etapa universitaria. Etapa que debo reconocer no ha sido fácil, pero que sin lugar a dudas habría sido mucho más difícil sin el apoyo y la ayuda de varias personas a quien me gustaría hacer mención en este espacio.

Primero quiero agradecer a la Universidad Nacional de La Pampa, y especialmente a la Facultad de Ingeniería, al personal docente y no docente, por darme la posibilidad de poder llevar a cabo los estudios en esta casa, y por el profesionalismo y la buena predisposición que ponen para lograr este objetivo.

Doy gracias a todos los estudiantes que me han acompañado en este camino, que han sido varios, con los que he pelado y tirado juntos para el mismo lado y de quienes me van a quedar buenos recuerdos durante mucho tiempo. Agradezco especialmente a Hernán y Nahuel, con quienes he pasado la mayor parte de la carrera, y el destino quiso que estemos juntos para concretar este proyecto de tesis; y quienes han sabido soportarme y ayudarme siempre.

Tengo un profundo agradecimiento a Abel Crespo, quien ha sabido incentivarme en los temas académicos que le compete, permitiéndome crecer profesionalmente y formarme en este área, y quien me orientó a relizar este proyecto de tesis, depositando una gran confianza en todo momento y siendo de mucha ayuda durante el desarrollo del mismo.

Agradezco especialmente a mi familia, por quienes tengo mucho afecto, y que han hecho posible que yo haya llegado hasta esta instancia. Un especial agradecimiento a mi papá Julio, quién me incentivó a inclinarme por este área de estudio y siempre me alentó a seguir adelante. También agradezco a mi mamá Mary, que ha sido un gran apoyo siempre y particularmente en los momentos más difíciles. Y mis hermanos, María y Martín, quienes han sido de aliento para que yo siga adelante.

Finalmente agradezco al resto de amigos y amigas, a quienes aprecio mucho, y que si bien no se involucraron directamente en los aspectos académicos, han sido y siguen siendo de especial compañía y su confianza siempre me ha dado fuerzas en todo lo que llevo adelante.

A todas estas personas que han contribuido de una u otra forma para que yo llegue al final de esta carrera universitaria, gracias!

Fredes, Marcos A.

Agradecimientos

Mediante este trabajo final de carrera concluyen muchos años de esfuerzos, alegrías, muchas horas de preocupación, estudio, tensiones, éxitos y fracasos. No solo es el fin de una etapa larga, sino también es el inicio de nuevos caminos que van a estar marcados por el orgullo y la sensación de haber alcanzado uno de los logros más importantes de mi vida. En este momento tan especial e irrepetible quiero hacer mención y agradecimiento a las personas que sin dudas me han marcado y ayudado a ser quién soy.

Quiero agradecer a Abel Crespo, nuestro director de tesis y profesor en varias materias de nuestra carrera. Siempre nos ha brindado su apoyo, su confianza, nos ha llenado de un gran respeto y valoración a nuestros esfuerzos. Pero más importante, siempre ha sabido compartir y trasmitir sus conocimientos, sus errores y aciertos. Una persona sincera y de manos totalmente abiertas.

Quiero agradecer a mis compañeros de facultad, que han sido de apoyo en incontables ocasiones. Personas con las que hemos formado un grupo de estudio maravilloso y que hemos podido llevar esa responsabilidad a un punto más profundo logrando entablar una amistad que será para siempre e incondicional. Sin lugar a dudas, las anécdotas y momentos vividos serán recuerdos vivos que no serán olvidados. Agradezco profundamente a Marcos Fredes, compañero de tesis y con quién hice todas las materias de la carrera, nos hemos visto en los mejores y peores momentos, me llevo de él un gran aprecio y admiración. Daniel Pérez y Sofía Aguirre, con quienes también he compartido gran parte de la carrera, amigos infaltables a los que vimos unirse en amor. Pablo Paccapelo, excelente persona, sensible y de gran corazón, hemos convertido horas de estudio en risas, llantos, alegrías y aprendizaje.

Quiero agradecer a mis amigos del alma, Francisco, Mauro y Fabián. Sin dudas las mejores personas que he conocido en mi vida, cada uno de ellos con características únicas, amigos que a pesar de la distancia y el tiempo siempre están en mí como ejemplos a seguir. Siempre les estaré agradecido por la amistad que me han brindado, son parte de mí.

A mi hermano Hernán, con quién llevo compartiendo toda mi vida, uno de mis pilares. Juntos hemos vivido grandes momentos, y sin dudas gran parte de mí se la debo a él. Quiero agradecer a mi familia en general pero aún más a mis padres, quienes nos han brindado todo su amor, fuerza,

esfuerzo y jamás nos hicieron faltar nada. Esta tesis y este título son en especial suyo, sin ellos todo esto no sería posible.

Por último, quiero agradecer a mi novia Carolina dueña de mi corazón y de mis alegrías. Ella me enseño lo más importante de la vida, el amor, el tener un corazón de oro. Estoy profundamente enamorado y orgulloso de la persona que es y de la persona que soy yo a partir de haberla conocido.

A todas las personas mencionadas, y a las que no me hayan entrado, gracias por todo lo vivido, y por este logro que no solo es mío, es nuestro.

Mora, C. Nahuel

Agradecimientos

Quiero agradecer profundamente a mi familia, ya que el apoyo que me brindan en cada uno de mis emprendimientos es fundamental para que pueda seguir avanzando y creciendo como persona, que conocen de mis esfuerzos para conseguir todo lo que logré y lo disfrutan conmigo. A mis padres, Néstor y Susana, por haberme dado todo lo que era necesario para que pueda obtener una buena educación y mis hermanos, Adrián, Analía y Nahuel, que aprecio todas y cada una de sus palabras de aliento en los momentos que me fueron necesarios.

A quien sabe, que siempre sentí a mi lado y también me acompañó en este trayecto final. Que con mucha paciencia me supo comprender ante mis dificultades y me brindó un gran apoyo, y por ello siempre estará en mi corazón.

A mis compañeros de trabajo final Nahuel y Marcos, hermano y amigo respectivamente, que desde el inicio de la carrera hemos conformado un lazo de amistad y de trabajo consolidado, donde cada uno complementó al otro y nos permitió avanzar hasta este momento.

A nuestro director, Abel Crespo, por darnos la oportunidad de formar parte del proyecto que nos permitió desarrollar el trabajo final de la carrera, depositándonos su confianza y dándonos total libertad para investigar como para utilizar todos los recursos que nos fueron necesarios dentro del laboratorio de redes.

A mis compañeros de carrera, que son parte de mi formación, y los amigos que me dejó para toda la vida, como son Sofía, Daniel y Elio.

Quiero agradecer también a todos aquellos que participaron de este trabajo de forma directa o indirecta ya que con su aporte fue posible desarrollar este trabajo.

A personal docente y no docente que conforma a la Facultad de Ingenería de la Universidad Nacional de La Pampa, por llevar adelante una tarea difícil como lo es la formación profesional de todos aquellos que ingresan a esta casa de estudios.

A todos ellos estaré siempre agradecido.

Mora, Hernán A.

Índice

Capítulo 1: Introducción y objetivos

1.1. Introducción

No hace mucho tiempo era impensado que las personas pudieran estar informadas minuto a minuto acerca de lo que pasa en cada rincón del mundo como lo es en la actualidad. Esto se debe a que la comunidad científica logra cada vez más avances significativos para la unión e intercambio de conocimientos. Sin dudas, un invento que marcó a la era actual fue el teléfono, que permitió que las personas pudieran estar comunicadas de una forma más rápida y constante a cómo estaban acostumbradas.

Las necesidades de las personas marcan la tendencia en la tecnología, ya que ésta evoluciona para satisfacerlas. Sin embargo, una vez satisfechas surgen otras, generando nuevos requerimientos e implicando la necesidad de seguir mejorando y creando nuevas tecnologías. Esto es lo que está ocurriendo actualmente en la telefonía tradicional.

El nacimiento de Internet marcó un punto de inflexión en los medios de comunicación y en la vida cotidiana. Hoy en día son más los dispositivos que tienen conectividad a Internet permitiendo funcionalidades innovadoras. Los distintos medios de comunicación como radios, diarios o televisión están adaptando estas nuevas tecnologías sobre Internet para extenderse y funcionar en este medio. La telefonía también está haciendo uso de este medio a través de voz sobre IP (VoIP). De esta forma, Internet se está convirtiendo en una red de propósito general en el cual todas estas formas de comunicación convergen y se está dejando de lado la necesidad de utilizar una red separada o dedicada para tal fin.

Actualmente, los sistemas telefónicos utilizan centrales de conmutación analógicas costosas, cuya ampliación o mantenimiento requiere de una mayor inversión. Los sistemas de licencia gratuita, como lo es GNU Linux, el software de código abierto y las comunidades de usuarios que apoyan a estas iniciativas, permiten el desarrollo de centrales telefónicas VoIP de menor costo dando una alternativa a las grandes corporaciones telefónicas actuales.

Un ejemplo de ello es Asterisk, la cual le permite a una organización poder implementar una central telefónica sin la necesidad de contar con hardware específico de gran potencia y evitar así

caer en una solución propietaria, lo que suele implicar un alto costo y una dependencia a su fabricante. Si bien Asterisk fue desarrollado para ser ejecutado sobre múltiples plataformas, la elección más atractiva es utilizarlo sobre GNU/Linux ya que se puede agregar nuevas funcionalidades sin tener que pagar nuevas licencias o, en el peor de los casos, reemplazar el equipamiento completo.

La portabilidad de los distintos servicios antes mencionados sobre Internet genera diferentes flujos de tráfico de paquetes. Debido a la heterogeneidad de estos flujos de datos, surgen distintos requerimientos de calidad para cada uno de ellos, lo que lleva a buscar una nueva forma de privilegiar o priorizar los recursos de la red. Es así como evolucionan los protocolos de red y transporte para poder satisfacer estas necesidades. Un ejemplo de esto es la utilización del campo de servicios diferenciados (DiffServ) en IPv4 y la adaptación de los protocolos de la capa de transporte, como por ejemplo TCP, para el soporte de notificación de congestión explicita (ECN), para evitar la pérdida de datos en la red.

1.2. Motivación

Las oportunidades de diseñar, desarrollar e implementar centrales telefónicas VoIP por parte de distintas entidades se han incrementado debido a la evolución de la tecnología, la facilidad de acceso a las redes de alta velocidad y la continua expansión de las mismas, además de la reciente desregularización de los servicios de Telefonía e Internet. No hay que olvidar que estas oportunidades sólo estaban disponibles para las grandes corporaciones telefónicas que concentran a la mayoría de los usuarios del mercado.

Hoy en día, el software libre y sistemas operativos GNU Linux achican la brecha entre las grandes corporaciones y los pequeños emprendedores, dando lugar a nuevas alternativas y generando competitividad en estos ámbitos.

Bajo estas consideraciones, la motivación de este Trabajo Final es el diseño, desarrollo e implementación de una central telefónica VoIP a bajo costo manteniendo la calidad de servicio en escenarios que abarcan la atención de grandes cantidades de usuarios.

1.3. Objetivos

Los objetivos pretendidos alcanzar durante este Trabajo Final son los siguientes:

1. **Diseño:** Diseñar una central telefónica VoIP adaptable a distintos ámbitos de aplicación en función de la capacidad de llamadas concurrentes que debe soportar. Se trata de que un conjunto de servidores VoIP interactúen según una política predefinida (balanceo de carga) y que el conjunto se comporte como una única central VoIP cuya capacidad es proporcional al número de servidores en el conjunto.

2. **Implementación**: Implementar una central VoIP Asterisk que permita brindar los servicios básicos de comunicación, y algunos adicionales tales como correo de voz e interconexión con redes Public Switched Telephone Network (PSTN), y abstraer la configuración de los servicios de modo tal que puedan ser reutilizados por otros servidores que brinden los mismos servicios o por subsistemas que faciliten su recolección.

3. **Calidad de Servicio:** Debe prestarse una calidad de servicio (QoS) a fin de evitar que los usuarios sean perjudicados por la elección de esta tecnología, ya que esperarán que el nuevo servicio iguale o supere al servicio tradicional. Para ello es imprescindible que no existan retrasos ni pérdidas en la comunicación. Se hará uso de las capacidades que tienen los protocolos de red y de transporte para brindar calidad de servicio, como por ejemplo servicios diferenciados (DiffServ).

4. **Interconexión:** Lograr la interconexión de servidores Asterisk en un entorno distribuido de redes privadas sobre Internet, mediante la utilización del protocolo IAX (Inter-Asterisk eXchange protocol).

5. **Alta disponibilidad:** Para que todo lo anterior tenga sentido debe implementarse un esquema de redundancia en el funcionamiento de la central telefónica a los efectos de evitar interrupciones de servicio originadas por distintos factores.

En pos de estos objetivos, en el capítulo 2 se verá una breve reseña de la evolución de la telefonía, algunos aspectos sobre el protocolo de internet (IP). Luego se verán algunas técnicas de la calidad de servicios, aplicables en las capas de red y de transporte del modelo de interconexión de sistemas abiertos (open system interconnection, u OSI), tales como la arquitectura de servicio diferenciado y la notificación de congestión explícita. También en este capítulo se presentarán aspectos de seguridad aplicables en internet como lo es el protocolo IPSec. El capítulo 3 presentará herramientas para la calidad de servicio en sistemas GNU/Linux, explicando primero como el tráfico es procesado y enviado a través del Kernel de Linux. También se verán técnicas de control de tráfico en el sistema operativo mencionado. El capítulo 4 explicará algunos protocolos de señalización necesarios para que los actores principales de la comunicación se puedan mantener comunicados, estos protocolos son los encargados de establecer las llamadas. El capítulo 5 presentará herramientas necesarias para dotar de alta disponibilidad y escalabilidad a cualquier infraestructura bajo la arquitectura GNU/Linux, en particular se explicará cómo aplicarlas a una central telefónica Asterisk. Los capítulos 6 y 7 comprenden el aspecto práctico de este trabajo implementando todos los conceptos y herramientas vistas hasta aquí vistas. También se presentará la topología final de este trabajo el cual reúne todos los objetivos anteriormente explicados. Finalmente se arribará a las conclusiones producto de las pruebas realizadas en los capítulos 6 y 7.

Capítulo 2: Calidad de Servicio (QOS)

2.1. Introducción

El estudio de los métodos y técnicas para la implementación de calidad de servicio en una red de paquetes conmutados es primordial para que la telefonía IP logre el éxito frente a la telefonía convencional. Es fundamental lograr igualar y mejorar las características que esta última ofrece para poder ser competitiva y atraer a las personas a esta nueva corriente. Para esto, en este capítulo se hará una breve referencia histórica de la evolución de la telefonía a través del tiempo hasta la actualidad. Luego, se verá como Internet y su infraestructura permite brindar soporte a distintos servicios, tales como telefonía, televisión, y otros servicios multimedia y en tiempo real. Todos estos avances generan nuevos tráficos de paquetes a través de Internet con distintas necesidades de servicio y prioridad, dando surgimiento a nuevas técnicas de calidad de servicio como servicios diferenciados y notificación de control de congestión para evitar la pérdida de información a través de la red. A partir de esto, se hará una pequeña reseña del protocolo IP y de las modificaciones que fueron necesarias realizar para adaptar el protocolo al transporte de estos nuevos datos, y finalmente se describirán las técnicas de calidad de servicios surgidas.

2.2. Evolución de la telefonía

Los primeros teléfonos no tenían una infraestructura de red sino que eran utilizados de forma privada y cableados de a pares. Es decir, los usuarios que querían hablar con diferentes personas tenían tantos pares telefónicos como fuese necesario para este propósito. Además, en un principio no poseían un medio de notificación de recepción de llamadas, para ello los usuarios que querían iniciar una comunicación debían silbar para alertar al otro extremo. Luego, este mecanismo fue reemplazado por una campana.

A medida que crecía la cantidad de usuarios, la complejidad de interconexión entre los mismos aumentaba. Para reducir dicha complejidad los teléfonos aprovecharon el principio de intercambio ya empleado en las redes de telégrafos. Cada teléfono fue cableado a una central telefónica, que a su vez era cableada a otras centrales. Todo esto fue realizado en forma jerárquica abarcando ciudades, países, continentes y océanos dando comienzo a lo que hoy en día se conoce como

Public Switched Telephone Network (PSTN).

En el comienzo de las centrales telefónicas, estas funcionaban de forma manual. Los operadores eran los encargados de interconectar a los usuarios para que estos pudieran comunicarse. De esta forma, una comunicación podía involucrar a varios operadores y tardar varios minutos hasta que pueda realizarse una llamada. Con el tiempo las centrales telefónicas fueron evolucionando y pasaron de manuales a semi-automáticas, luego a automáticas, posteriormente a electrónicas y finalmente a centrales telefónicas IP, tecnología que en algunos países todavía está emergiendo.

La tendencia actual es transmitir las comunicaciones de voz sobre redes que utilizan el protocolo de Internet, esto se conoce también como VoIP. La telefonía de Internet se refiere a los servicios de comunicación que son transportados a través de Internet en lugar de utilizar la red telefónica tradicional (PSTN). Los pasos involucrados para originar una llamada VoIP son señalización, configuración del canal de comunicación, digitalización de la señal de voz analógica, compresión opcional, empaquetado y transmisión como paquetes IP a través de la red de paquetes conmutados. En el receptor se llevan a cabo pasos similares para reproducir el audio original.

Los sistemas VoIP emplean protocolos de control de sesión para controlar el montaje y desmontaje de las llamadas como así también codificadores de audio que permiten su transmisión a través de Internet como sonido digital. Existen distintas implementaciones, algunas priorizan la calidad mientras que otras optimizan el ancho de banda utilizado.

Las comunicaciones en la red IP son inherentemente menos confiables en comparación con la red telefónica convencional, y por lo tanto no provee un mecanismo para asegurarse que los paquetes no sean extraviados, o entregados en el orden correcto. Esta es una red de mejor esfuerzo sin garantías fundamentales de Calidad de Servicio. Por lo tanto, las implementaciones de VoIP pueden enfrentar problemas mitigando latencia y jitter.

Por defecto, los routers manejan el tráfico de forma FIFO (primero en entrar, primero en salir). Los routers que procesan grandes volúmenes de tráfico pueden producir latencias que superan los umbrales permitidos para VoIP. Los retrasos fijos no pueden ser controlados ya que son causados por las distancias físicas que los paquetes deben viajar, sin embargo, la latencia puede ser minimizada haciendo que los paquetes de voz sean priorizados a través de mecanismos tales como

DiffServ (Servicios Diferenciados, sección 2.4).

La mayoría de las implementaciones VoIP son compatibles con el estándar de numeración global E.164, permitiendo así poder realizar llamadas desde teléfonos de VoIP a teléfonos fijos (PSTN) o móviles (PLMN), logrando integración con la telefonía tradicional.

2.3. Protocolo IP

El Protocolo de Internet (Protocolo IP) fue diseñado para ser usado en sistemas de redes de comunicación de intercambio de paquetes, como lo eran las *catenet*[1]. El Protocolo IP proporciona los medios necesarios para la transmisión de datagramas (bloques de datos en la capa de red) desde el origen de la comunicación al destino de la misma, donde origen y destino son hosts identificados por direcciones de longitud fija, llamadas direcciones IP. También, si es necesario, el Protocolo IP se encarga de la fragmentación y el re-ensamblaje de grandes datagramas para su transmisión a través de redes de trama pequeña. En síntesis, lleva a cabo dos funciones básicas: *direccionamiento* y *fragmentación*.

En cada host y en las pasarelas que interconectan redes reside un módulo del Protocolo IP que está involucrado en la operación de la comunicación. Estos módulos utilizan las direcciones IP que se encuentran en el encabezado del protocolo para transmitir los datagramas hacia sus destinos. A la selección de un camino para la transmisión de datos se la denomina encaminamiento y se presenta especialmente en las pasarelas. Además, estos módulos usan otros campos en el encabezado para fragmentar y re-ensamblar los datagramas cuando sea necesario para su transmisión a través de redes de trama pequeña.

El Protocolo IP trata a cada datagrama como una entidad independiente no relacionada con ningún otro datagrama, por lo tanto, no existen conexiones o circuitos lógicos (virtuales o de cualquier otro tipo) para la transmisión de estos. El protocolo, además, utiliza cuatro mecanismos clave para prestar su servicio: Tipo de Servicio (*ToS – Type of service*), Tiempo de Vida (*TTL – Time To Live)*, Opciones (*Options*), y Suma de Control de Encabezado (*Header Checksum*).

[1]Es un término obsoleto para un sistema de redes de comunicación de intercambio de paquetes a través de gateways.

El Tipo de Servicio se utiliza para indicar la calidad del servicio deseado. Está conformado de un conjunto abstracto o generalizado de parámetros que caracterizan las elecciones de servicio presentes en las redes que forman Internet. Esta indicación de tipo de servicio será usada por las pasarelas para seleccionar los parámetros de transmisión efectivos para una red en particular, la red que se utilizará para el siguiente salto, o la siguiente pasarela al encaminar un datagrama.

El Tiempo de Vida es una indicación de un límite superior en el periodo de vida de un datagrama. Es fijado por el emisor del datagrama y reducido en los puntos a lo largo de la ruta donde es procesado. Si el tiempo de vida se reduce a cero antes de que el datagrama llegue a su destino, será destruido. Puede verse al tiempo de vida como en un plazo de autodestrucción para evitar que el datagrama quede atrapado en la red en encaminamientos infinitos.

Las Opciones proporcionan funciones de control necesarias o útiles en algunas situaciones pero innecesarias para las comunicaciones más comunes. Las opciones incluyen recursos para marcas de tiempo, seguridad y encaminamiento especial.

La Suma de Control de Encabezado proporciona una verificación de que la información utilizada al procesar el datagrama ha sido transmitida correctamente. Los datos pueden contener errores. Si la suma de control de cabecera falla, el datagrama es descartado inmediatamente por la entidad que detecta el error.

El Protocolo IP no proporciona ningún mecanismo de comunicación fiable. No existen acuses de recibo entre extremos ni entre saltos. No hay control de errores para los datos, sólo una suma de control de cabecera. No hay retransmisiones y no existe el control de flujo. Los errores detectados pueden ser notificados por medio del Protocolo de Mensajes de Control de Internet (*ICMP - Internet Control Message Protocol)* el cual está implementado en el módulo del Protocolo IP.

2.3.1. Encabezado del Protocolo IPv4

A continuación se muestra un resumen del encabezado del Protocolo IP (Figura 2.3.1):

Figura 2.3.1: Encabezado Protocolo IPv4.

Versión: Indica el formato del encabezado IP.

IHL: Indica la longitud del encabezado IP en palabras de 32 bits, de forma que indica el comienzo de los datos (*payload*). Se puede observar que el mínimo valor para un encabezado correcto es 5.

Tipo de servicio (ToS): Provee una indicación de los parámetros abstractos de la calidad de servicio deseada. Estos parámetros son utilizados para guiar la selección de los parámetros de servicio reales cuando el datagrama es transmitido a través de una red en particular.

Algunas redes ofrecen prioridad de servicio, la cual trata al tráfico de más alta prioridad como más importante que el resto del tráfico (generalmente aceptando sólo tráfico por encima de cierta prioridad en momentos de sobrecarga). La elección más común es un compromiso de tres niveles entre Baja Demora (*Low-Delay*), Alta fiabilidad (*High-Reliability*), y Alto Rendimiento (*High-Throughput*).

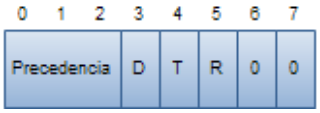


Figura 2.3.2: Bits del campo ToS.

La interpretación de los bits en el campo ToS (Figura 2.3.2) es la siguiente:

- **Bits 0-2:** Precedencia.
  - 111 - Control de Red
  - 110 - Control Entre Redes
  - 101 - CRITICO/ECP
  - 100 - Muy urgente (*Flash Override*)
  - 011 - Urgente (*Flash*)
  - 010 - Inmediato
  - 001 - Prioridad
  - 000 - Rutina
- **Bit 3:** 0 = Demora Normal, 1 = Baja Demora.
- **Bit 4:** 0 = Rendimiento Normal, 1 = Alto rendimiento.
- **Bit 5:** 0 = Fiabilidad Normal, 1 = Alta fiabilidad.
- **Bits 6-7:** Reservado para uso futuro.

Longitud total: Es la longitud del datagrama, medida en octetos, incluyendo el encabezado IP y los datos. Este campo permite una longitud máxima de datagrama de 65.535 octetos.

Identificación: Este valor de identificación es asignado por el emisor para permitir el ensamblado de los fragmentos de un datagrama.

Banderas: Existen varias banderas de control (Figura 2.3.3).

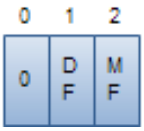


Figura 2.3.3: Banderas de control.

- **Bit 0:** Reservado, debe ser cero.
- **Bit 1:** No Fragmentar (*DF - Don't Fragment*)
  - 0 = Puede fragmentarse, 1 = No fragmentar.
- **Bit 2:** Más Fragmentos (*MF - More Fragments*)
  - 0 = Último fragmento, 1 = Más fragmentos.

Posición del fragmento: Este campo indica a que parte del datagrama original pertenece un fragmento. La posición del fragmento se mide en unidades de ocho octetos (64 bits). El primer fragmento tiene posición cero.

Tiempo de vida: Indica el máximo tiempo que el datagrama tiene permitido permanecer en la red. Si este campo tiene valor cero, entonces el datagrama debe ser destruido. Cada módulo que procese un datagrama debe disminuir el valor de tiempo de vida por lo menos una unidad. La intención es hacer que los datagramas imposibles de entregar sean descartados, y limitar el máximo periodo de vida de un datagrama.

Protocolo: Este campo indica el protocolo usado en la capa de transporte.

Suma de Control de Encabezado: Se realiza una suma de control del encabezado y no de todo el datagrama. Dado que algunos campos de la cabecera cambian (por ejemplo, el tiempo de vida), esta suma es recalculada y verificada en cada punto donde es procesado el encabezado. El algoritmo a utilizar es el complemento A1 de 16 bits de la suma de los complementos A1 de todas las palabras de 16 bits de la cabecera. A la hora de calcular la suma de control, el valor inicial de este campo es cero.

Dirección origen: Es la dirección IP del emisor del datagrama. Esta dirección tiene un tamaño fijo de 32 bits.

Dirección destino: Es la dirección IP del destinatario del datagrama. Esta dirección tiene un tamaño fijo de 32 bits.

Opciones: Es un campo de longitud variable que puede aparecer o no en los datagramas.

Relleno: El valor de relleno se utiliza para asegurar que el tamaño del encabezado IP es un múltiplo de 32 bits. Este relleno se completa con ceros.

2.3.2. Definiciones históricas del octeto ToS en IPv4

En la evolución del Protocolo IP, el campo con mayor número de modificaciones durante este proceso es el campo ToS (*Type of Service*). Estos cambios son reflejados en los distintos *Request for comments* (*RFC*) que los trata.

En el RFC 791 (Information Sciences Institute - University of Southern California, 1981), la disposición de los bits (en la Figura 2.3.4) es la siguiente: Los bits cero a dos son destinados al campo Precedencia (*Precedence*), los bits tres al cinco pertenecen al campo TOS, mientras que los bits seis y siete fueron "*reservados para uso futuro*".

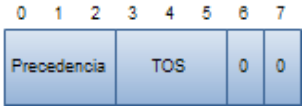


Figura 2.3.4: Octeto ToS definido por el RFC 791.

En el RFC 1122 (Braden, 1989), los bits 6 y 7 (como se ve en la Figura 2.3.5) fueron incluidos al campo TOS, aunque no se especificó ningún uso para dichos bits.

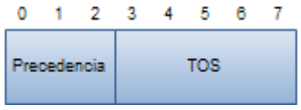


Figura 2.3.5: Octeto ToS definido por el RFC 1122.

Más adelante, en el RFC 1349 (Almquist, 1992), el bit 6 en el campo TOS fue definido para minimizar el costo monetario (*MinimizeMonetaryCost*) y en adición al campo TOS y al campo Precedence se agregó un último campo llamado MBZ (must be zero) como se ilustra en la Figura 2.3.6, el cual fue definido como actualmente sin uso. El RFC 1349 (Almquist, 1992) declara que el emisor de un datagrama debe establecer el campo MBZ en cero a menos que este participando en un protocolo de Internet experimental que haga uso de dicho bit.

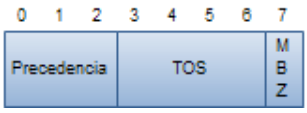


Figura 2.3.6: Octeto ToS definido por el RFC 1349.

El RFC 1455 (Eastlake, 1993), define un estándar experimental que utiliza los cuatro bits del campo TOS para solicitar un nivel de seguridad en el enlace.

Los documentos RFC 1349 (Almquist, 1992) y RFC 1455 (Eastlake, 1993) quedaron obsoletos por el RFC 2474 (K. Nichols, 1998) denominado "*Definition of the Differentiated Services Field (DS Field)*

in the IPv4 and IPv6 Headers". En él se define dos nuevos campos en el octeto ToS (verFigura 2.3.7). El campo DSCP (*Differentiated Services Codepoint*) que va del bit cero al bit cinco, y el campo CU (*Currentlyunused*) en los bits seis y siete. La finalidad de estos campos se explicará en el apartado *Servicios Diferenciados* de este capítulo.

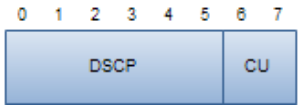


Figura 2.3.7: Octeto ToS definido por el RFC 2474.

El RFC 2780 (S. Bradner, 2000), define un uso experimental para los últimos dos bits del campo ToS (Figura 2.3.8), el cual sirve para la notificación de congestión explicita (ECN). La finalidad de estos bits se explicará en el apartado *Notificación de congestión explícita.*

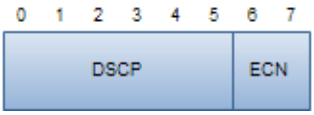


Figura 2.3.8: Octeto ToS definido por el RFC 2780.

Como se puede observar, el octeto ToS del encabezado IPv4 ha sufrido grandes cambios a lo largo del tiempo. Estos cambios hacen que no exista compatibilidad entre los distintos nodos o routers que sean compatibles con uno u otro RFC. A la hora de montar un escenario para implementar alguno de los usos que ha recibido el octeto ToS, se deberá considerar esta característica en los routers de nuestra red.

2.4. Servicios diferenciados (DiffServ)

Los servicios diferenciados surgen ante la necesidad de brindar una calidad de servicio (QoS – Quality of Service) a distintos tipos de flujos de datos (tales como voz, video, música, páginas web, correo electrónico, etc.) con diferentes requerimientos de calidad.

En diciembre de 1998, la IETF (Internet Engineering Task Force) publicó el RFC 2474 (K. Nichols, 1998) en el cual se reemplaza al campo ToS del encabezado IP por el campo DiffServ (DS field). DiffServ surgió como una alternativa a IntServ (Integrated Services) el cual había sido propuesto y

publicado en el RFC 1633 (R. Braden, 1994). La mayor ventaja de DiffServ sobre IntServ es que no necesita de protocolos de señalización. El modelo IntServ usa un protocolo de señalización que debe funcionar tanto en los host como en los routers. Si la red tiene miles de flujos, los routers deben almacenar información de estado para cada uno de ellos que pasa a través de él. Esto es un problema serio de escalabilidad que ha hecho que IntServ no se haya convertido en un modelo popular. Por otro lado, se puede observar que IntServ permite una clasificación de servicio más refinada que DiffServ ya que almacena información para cada uno de los flujos en particular mientras que DiffServ está centrado a clases de flujos.

El campo *DS* (Figura 2.3.7) está conformado por ocho bits, situados en los bits 8 al 16 del encabezado en IPv4 y del 4 al 12 en el encabezado IPv6 (en el campo *Traffic Class*). Los primeros seis bits del campo se utilizan como un *codepoint* (*DSCP*) para asignar el comportamiento del paquete en cada salto (*Per-hop Behavior o PHB*). Los valores que este puede contener no están totalmente estandarizados, lo que permite definir nuevos comportamientos por salto que se adecúen a los propósitos específicos de un dominio en particular. El propósito de los dos últimos bits del campo *DS*está definido en el RFC 3168 (K. Ramakrishnan, 2001) y se utiliza para la notificación de congestión explícita. Sin embargo, al momento de elegir el comportamiento que debe aplicarse a los paquetes recibidos, los nodos que implementan servicios diferenciados ignoran estos dos últimos bits.

Al implementar servicios diferenciados se debe dar soporte a la lógica necesaria para la configuración de la correlación o mapeo entre los *codepoints* y *PHBs*, es decir, que a cada valor de *DSCP* se debe asignar un comportamiento. Esta configuración debe incluir un *codepoint* recomendado por defecto, que debe ser único. Aquellos paquetes que son recibidos con un *codepoint* desconocido deben ser reenviados como si estuviesen marcados con el comportamiento por defecto, y sus *codepoints* no deberían ser modificados.

2.4.1. Dominio DS

Un dominio que implementa servicios diferenciados se lo denomina *Dominio DS o Dominio DiffServ* (Figura 2.4.1). Este dominio se compone de nodos extremos, que se encuentran en el borde del dominio; y nodos internos, conformando el núcleo del dominio. Los nodos extremos, a

su vez, se clasifican en dos tipos, como lo describe el RFC 2475 (S. Blake, 1998):

- **Nodos de Ingreso (DS Ingress):** Encargados de clasificar, acondicionar el tráfico de entrada, cumplir con los TCA[2] (*Traffic Conditioning Agreement*) establecidos entre dominios DS y cumplir con el Acuerdo de Nivel de Servicio[3] *(SLA)*, elcual se establece entre los usuarios y el proveedor del Dominio DS. La clasificación puede estar en función de dirección IP y puerto (tanto origen como destino), protocolo de transporte y DSCP. Este tipo de clasificador se conoce como MF (Multi-Field Classifier).

- **Nodos de Salida (DS Egress):** Pueden realizar funciones TCA/SLA para el tráfico que se encamina hacia otros dominios o usuarios, remarcando los paquetes si es necesario.

Los nodos internos son los encargados de encaminar los flujos de tráfico según sus niveles de prioridad dentro del dominio. A diferencia de los nodos extremos, para la selección del PHB el clasificador solo debe tener en cuenta el campo DSCP. El clasificador de este tipo es conocido como BA (Behavior Aggregate Classifier).

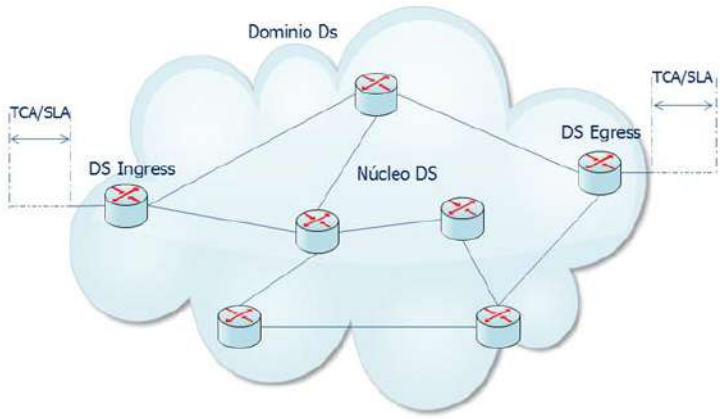


Figura 2.4.1: Dominio DiffServ.

[2] Acuerdo que especifica las reglas de clasificación, y correspondientes perfiles de tráfico, marcado, descarte, y adaptación aplicables a los flujos de tráfico seleccionados por el clasificador.
[3] Contrato entre un usuario y el proveedor de servicios diferenciados que especifica el tipo de servicio que recibirá el usuario. Puede incluir reglas de acondicionamiento de tráfico, que constituyen el TCA.

2.4.2. Arquitectura de los nodos DS

Los nodos extremos de un dominio DS son los más importantes dado que clasifican y permiten el ingreso de tráfico a la red. También, pueden acondicionar tráfico para asegurarse que se satisfagan los requerimientos de políticas. Los nodos extremos contienen los siguientes elementos (Figura 2.4.2):

- **Clasificador (Classifier):** Se pueden distinguir dos tipos de clasificadores:
  - **Clasificador Standard (Standard Classifier):** También denominado como Clasificador BA (*Behavior Aggregate Classifier*). Selecciona los paquetes en base al valor DSCP.
  - **Clasificador Multi-Field (Multi-Field Classifier):** Selecciona los paquetes en base a uno o más campos del encabezado IP, tales como dirección origen/destino, protocolo, puerto origen/destino, etc.

- **Marcador (Marker):** Es una entidad que se encarga de establecer el campo DSCP.

- **Sistemas de política (Policy Systems/Bandwidth Brokers):** Son dispositivos configurados con las políticas de la organización. Llevan un control de las asignaciones actuales del tráfico marcado e interpreta nuevas peticiones para marcar el tráfico en función de las políticas y las asignaciones actuales.

- **Acondicionador de tráfico (Traffic Conditioner):** Es una entidad que mide, marca, descarta y da forma al tráfico. El acondicionador de tráfico puede remarcar un flujo de tráfico, o puede descartar o dar forma a los paquetes para alterar las características temporales del flujo y hacerla compatible a un perfil de tráfico. Un perfil de tráfico especifica las propiedades temporales de un flujo de tráfico seleccionado por el clasificador y es un componente opcional de un TCA. Los subcomponentes del acondicionador de tráfico son el marcador, el medidor, el descartador, el adaptador y el impositor de políticas (policer).

- **Medidor (Meter):** Mide la tasa de flujos de tráfico seleccionadas por el clasificador. Las

mediciones son usadas por los siguientes elementos o para propósitos de medición y contabilización.

- **Impositor de políticas (Policer):** Evalúa las mediciones hechas por el medidor y las utiliza para aplicar perfiles de tráfico basado en políticas.

- **Descartador (Dropper):** Se encarga de descartar algunos o todos los paquetes en un flujo de tráfico para lograr compatibilidad con el perfil de tráfico.

- **Adaptador (Shaper):** Retrasa paquetes dentro de un flujo de tráfico en conformidad a algún perfil de tráfico definido. El adaptador puede descartar paquetes si no posee suficiente espacio en el buffer para almacenar los paquetes retrasados.

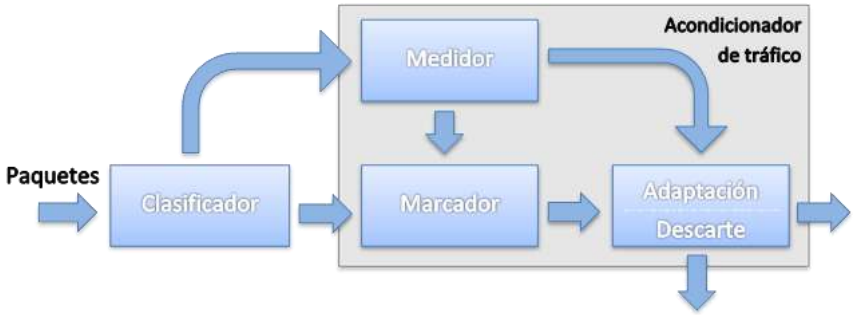


Figura 2.4.2: Arquitectura de un nodo extremo.

En contraste a esta arquitectura, la arquitectura de los nodos internos del Domino DS (Figura 2.4.3) son más simples debido a que únicamente se encargan de clasificar los paquetes en base a la interpretación del valor del campo DSCP y despacharlos de acuerdo al PHB asignado.

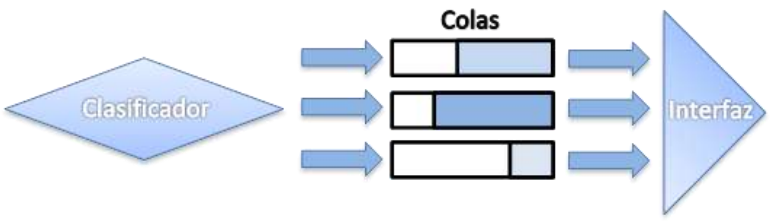


Figura 2.4.3: Arquitectura de un nodo interno.

2.4.3. Per-hop Behavior

Las PHBs son las descripciones de los comportamientos externamente observables de reenvío, aplicadas a un DS behavior aggregate en un nodo DS. Un DS behavior aggregate es una colección de paquetes con el mismo codepoint DS que atraviesa un enlace en una dirección en particular. En otras palabras, la PHB es el medio por el cual un nodo reserva recursos para los DS behavior aggregates.

Las PHBs pueden ser especificadas en términos de prioridad de recursos (por ejemplo, buffer, ancho de banda, etc.) relativos a otras PHBs, o en términos de características relativas observables de tráfico (por ejemplo, retraso, perdida de paquetes, jitter, etc.). Estas PHBs pueden ser usadas como bloques de construcción para alojar recursos y deberían ser especificadas como un grupo (PHB group) para lograr consistencia. Los PHB groups generalmente compartirán una restricción en común aplicada a cada una de las PHB dentro del grupo, tales como planificación de paquetes o políticas de administración de buffer. Una única PHB puede ser vista como un caso especial de un grupo PHB con un único miembro.

La selección de una PHB en un nodo se realiza a través de la evaluación del codepoint DS de los paquetes recibidos. Existen PHBs que han sido estandarizadas, las cuales poseen un codepoint recomendado. Sin embargo, el espacio total para codepoints es más grande que el espacio disponible para codepoints recomendados de las PHBs estandarizadas, lo cual deja lugar para la configuración de codepoints de forma local. Todos los codepoints deben ser mapeados a una PHB; si no existe alguna política local, los codepoints que no son mapeados a una PHB estandarizada deberían ser mapeadas a la PHB por defecto.

Las per-hop behavior pueden ser clasificadas en cuatro grupos:

- Default PHB -*PHB por defecto, RFC 2474* (K. Nichols, 1998)-
- Class Selector PHB Requirements -*RFC 2474* (K. Nichols, 1998)-
- Assured Forwarding (AF) PHB -*RFC 2597* (J. Heinanen, 1999)-
- Expedited Forwarding (EF) PHB -*RFC 2598* (V. Jacobson, 1999)-

2.4.3.1. PHB por defecto

Los nodos que implementan servicio diferenciado deben disponer de un comportamiento por defecto. Este comportamiento es generalmente conocido como comportamiento de mejor esfuerzo (*best-esffort*) definido en el RFC 1812 (Baker, 1995), donde los paquetes enviados no adhieren a ninguna regla en particular y la red enviará tantos paquetes y tan rápido como le sea posible. Una implementación razonable de este comportamiento podría ser una disciplina de cola que envía estos paquetes cuando el enlace de salida no es requerido para satisfacer otros comportamientos. Sin embargo, se debería asegurar que los paquetes no queden pospuestos indefinidamente. Este PHB por defecto permite a los emisores que no están en conocimiento del uso de servicio diferenciado seguir utilizando la red sin problemas. El *codepoint* recomendado para el *PHB* por defecto es el patrón de bits “000000”.

Para lograr calidad de servicio a través de distintos dominios DS, es posible establecer un acuerdo entre los dominios intervinientes para que paquetes marcados con el comportamiento por defecto puedan ser remarcados con otros codepoint al entrar a otro dominio DS.

2.4.3.2. Class Selector Codepoint y Class Selector PHB Requirements

Para proveer compatibilidad con el uso del campo IP Precedence, el RFC 2474 (K. Nichols, 1998) definió los términos Class Selector Codepoints y Class Selector PHB Requirements. El primer término se refiere al conjunto de codepoints cuyos valores se encuentran en el rango “XXX000” donde las X pueden ser “0”ó “1” independientemente, mientras que el segundo término se refiere a los PHBs a los cuales estos codepoints deben ser mapeados.

El valor numérico Class Selector Codepoint tiene un significado relativo. A mayor valor numérico, más alto el valor relativo. Por otro lado, el conjunto de PHBs que son mapeados por los Class Selector Codepoint deben ofrecer por lo menos dos clases de reenvío de tráfico independientes. De ésta forma, dados dos conjuntos de paquetes marcados con diferentes valores numéricos de Class Selector Codepoint, el PHB seleccionado debería brindar mayor prioridad a aquellos paquetes marcados con el Class Selector Codepoint de mayor valor relativo.

2.4.3.3. Assured Forwarding (AF) PHB Group

Es el medio por el cual un dominio DS del proveedor puede ofrecerle diferentes niveles de garantías de transmisión a paquetes IP recibidos desde el dominio DS de un cliente. Existen cuatro clases AF definidas, donde cada una de ellas se encuentra, en cada nodo, asignadas a una cierta cantidad de recursos de envío (espacio de buffer, ancho de banda, etc.). Cada una de estas clases es independiente entre sí y a su vez poseen tres niveles distintos de precedencia de descarte.

Dentro de cada clase AF, los paquetes son marcados con uno de los tres posibles valores de precedencia de descarte (*DropPrecedence*). En caso de congestión, la caída de precedencia de los paquetes determina la importancia relativa de los paquetes dentro de la clase AF. Un nodo congestionado intentará proteger del descarte de paquetes a aquellos que contengan un valor de precedencia de descarte menor, prefiriendo descartar a los paquetes con un valor mayor de precedencia de descarte. Así, en un nodo DS, el nivel de garantía de envío depende de:

- La cantidad de recursos de transmisión asignados a la clase AF al cual el paquete pertenece.

- La carga actual de la clase AF y, en caso de congestión, a la carga de los niveles dentro de la misma.

- La caída de precedencia del paquete.

Estas cuatro clases AF y sus tres niveles de precedencia de descarte están definidas para propósitos generales, pero pueden definirse más clases y niveles para uso local.

Un nodo DS debería implementar las cuatro clases de uso general. Los paquetes en una clase AF deben ser reenviados de forma independiente a los paquetes pertenecientes a otra clase AF. Además, debe reservar una cantidad mínima y configurable de recursos de envío para cada clase AF implementada.

Una implementación AF debe tratar de minimizar las congestiones a largo plazo entre cada clase y permitir congestiones a corto plazo como producto de ráfagas de paquetes. Esto requiere un algoritmo de administración de colas activo, tal como Random Early Drop (RED). También, debe

detectar y responder a congestiones de largo plazo dentro de cada clase mediante el descarte de paquetes, y manejar las congestiones de corto plazo mediante el encolamiento de paquetes.

El algoritmo de descarte debe tratar a todos los paquetes dentro de una misma clase y nivel de precedencia de la misma forma. Esto implica que para un nivel de congestión dado, la tasa de descarte para un micro flujo de paquetes en particular dentro de un único nivel de precedencia será proporcional al porcentaje de ese flujo con respecto a la cantidad total de tráfico pasando a través de dicho nivel.

El RFC 2597 (J. Heinanen, 1999) introduce los codepoints recomendados (Tabla 2.4.1) para utilizar las cuatro clases AF con sus respectivos niveles de precedencia de descarte. Estos codepoints no son utilizados por ningún otro grupo PHB definidos.

Precedencia de descarte	Clase 1	Clase 2	Clase 3	Clase 4
Baja	001010	010010	011010	100010
Intermedia	001100	010100	011100	100100
Alta	001110	010110	011110	100110

Tabla 2.4.1: Codepoints recomendados para AF PHB.

2.4.3.4. Expedited Forwarding (EF) PHB

Este tipo de PHBs pueden utilizarse para brindar, a través de dominios DS, servicios extremo-a-extremo disminuyendo pérdidas, latencia, jitter y dando ancho de banda asegurado. Este tipo de servicio es definido como *Servicio Premium*.

Las pérdidas, latencia y jitter se deben al encolado de los paquetes cuando son transmitidos en una red. Por lo tanto, proveer baja pérdida, latencia y jitter para algún conjunto de tráfico significa asegurar que ese conjunto nunca será encolado o será encolado en una cola pequeña. Las colas aparecen cuando la tasa de arribo de tráfico de un nodo es mayor que su tasa de salida. Así, un servicio que asegura que no se encola un conjunto de paquetes es equivalente a establecer tasas tales que en cada nodo por donde pasa el tráfico, la máxima tasa de arribo es menor a la tasa de salida.

Proveer un servicio de este tipo constaría de dos partes: La primera consiste en configurar los

nodos de modo que los conjuntos de tráficos tengan una tasa de salida bien definida, donde el significado de bien definida se refiere a la independencia del estado de los nodos y de la intensidad de otros tipos de tráfico que atraviesan a los mismos. La segunda parte consiste de condicionar los conjuntos de tráfico de manera que la tasa de arribo de paquetes en cualquier nodo es siempre menor que su tasa de salida configurada.

El EF PHB provee la primer parte del servicio, mientras que la segunda parte es provista por el acondicionador de tráfico de los nodos extremos del dominio DS descriptos anteriormente.

ExpeditedForwarding PHB no es una parte obligatoria de la arquitectura de servicios diferenciados, es decir, un nodo no tiene obligación de implementar EF PHB para ser considerado compatible con DS. Sin embargo, de implementarlo debe sí o sí satisfacer las especificaciones del RFC 2598 (V. Jacobson, 1999). Si EF PHB es implementado por mecanismos que permiten derecho de prioridad ilimitado a otros tráficos, la implementación debe incluir algún medio por el cual se limite el daño que el tráfico EF podría causar al resto del tráfico. El tráfico que excede este límite debe ser descartado. Esta tasa máxima EF, y si es necesario un tamaño de ráfaga de paquetes, debe ser configurable por el administrador de la red. La tasa máxima y mínima pueden ser la misma y ser configuradas por un solo parámetro.

Los paquetes marcados como EF PHB para un determinado Dominio DS deberían ser remarcados únicamente con otro codepoint que satisfaga EF PHB al ingresar al nuevo dominio en el borde del mismo. El codepoint recomendado para este tipo de PHB es 101110.

2.4.4. Espacio de CodePoints

El campo DSCP permite utilizar 64 codepoints distintos y estos están divididos en tres pools para diferenciar sus usos: un pool de 32 codepoints recomendados (Pool 1) para ser asignado para acciones estándar, un pool de 16 codepoints (Pool 2) para ser reservados para experimentación o uso local (EXP/LU), y un pool de 16 codepoints (Pool 3) que son inicialmente para experimentación o uso local, pero debería ser utilizado para asignaciones estándar si es que el Pool 1 está completo. Los pools se definen en la Tabla 2.4.2, donde las X pueden referirse a “0” ó “1”.

Pool	Espacio de Codepoint	Política de asignación
1	XXXXX0	Acciones estándar
2	XXXX11	Experimentación o uso local
3	XXXX01	Experimentación o uso local

Tabla 2.4.2: Espacio de codepoints divididos en pools.

Los codepoints recomendados ("XXX000") que fueron anteriormente mencionados son tomados del Pool 1. Estos codepoints deben ser mapeados a PHBs que brinden un mínimo nivel de compatibilidad con IP Precedence.

2.4.5. Robo y Denegación de servicio

La definición del estándar prevé el caso de que un atacante quisiera obtener un mejor servicio del que le corresponde mediante la modificación de los codepoints a valores que indican un comportamiento usado por paquetes de tráfico de mayor privilegio. Llevando este caso al extremo, este robo de servicio puede transformarse en un ataque de denegación de servicios cuando el tráfico modificado o inyectado agota los recursos disponibles.

La defensa para esta clase de ataque consiste en la combinación de control de tráfico en los límites del dominio DS con una infraestructura segura e íntegra de la red dentro del dominio. Los nodos del límite del dominio DS deben asegurar que todos los paquetes entrantes sean marcados con valores de codepoints adecuados al tráfico y al dominio, remarcando el tráfico con nuevos valores de codepoints de ser necesario. Estos nodos del límite son la primera línea de defensa frente a los ataques de robo y denegación de servicio. Los nodos en el interior del dominio DS confían en los codepoints para asociar el tráfico con las PHB, y no requieren verificar el valor de codepoint antes de usarlo. Como resultado, los nodos interiores dependen de la correcta operación de los nodos en el límite del dominio DS.

2.5. Notificación de congestión explícita

Otro aspecto a tener en cuenta en la calidad de servicio, que puede brindarse al tráfico de un dominio, es evitar la pérdida de paquetes por congestión en los routers. Para esto, es necesario lograr una compatibilidad entre la capa de transporte, en particular TCP, y la capa de red (IP). En este apartado del capítulo se tratará la Notificación de Congestión Explícita (ECN) en IP como una

alternativa al problema de congestión.

Uno de los mecanismos para administrar colas activas (AQM) que fueron propuestos para detectar el inicio de congestión es el *Random Early Detection* (RED). AQM está pensado para ser un mecanismo general utilizando una de las tantas alternativas para indicar congestión, pero en ausencia de ECN, está restringida al uso del descarte de paquetes como mecanismo para indicar la congestión. AQM descarta paquetes basándose en el promedio de longitud de la cola que excede un umbral, en lugar de basarse solo en el desbordamiento de la cola. Dado que AQM puede descartar paquetes antes de que la cola realmente se llene, no siempre está forzado a descartarlos debido a limitaciones de memoria.

Cuando el codepoint *Congestion Experienced* (CE) es provisto en el encabezado IP e interpretado por el protocolo de transporte, AQM puede establecer dicho codepoint en el paquete en vez de descartarlo. El uso de codepoint CE con ECN le permite al o los receptores recibir los paquetes, evitando el potencial retraso excesivo debido a la retransmisión de los mismos después de perderse. Un paquete marcado con el codepoint CE se denomina *paquete CE*.

2.5.1. Campo ECN y los valores que adopta

El campo ECN se encuentra en el encabezado IP en los bits 6, denominado *ECN-CapableTransport* (ECT), y 7, *Congestion Experienced (*CE), del campo DS, dando lugar a cuatro posibles valores de codepoints ECN (“00”, “01”, “10” y “11”). En la siguiente Figura se muestra el campo mencionado.

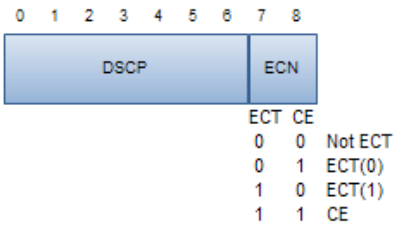


Figura 2.5.1: Bits ECN del campo DS.

Para indicar que los extremos del protocolo de transporte son compatibles con ECN, el emisor de la comunicación utiliza los codepoints “10” y “01”, llamados ECT(0) y ECT(1) respectivamente. Los routers tratan a estos codepoints equivalentemente, y los emisores son libres de usar cualquiera de estos dos valores para indicar compatibilidad en la capa de transporte con ECN en todos los

paquetes. El RFC 2481 (Ramakrishnan, 1999), definía únicamente el ETC(0) para indicar que los nodos extremos son compatibles con ECN, pero por cuestiones de seguridad este RFC quedó obsoleto por el RFC 3168 (K. Ramakrishnan, 2001) en el cual se especifican los codepoints recién mencionados. El uso de ambos codepoints ECT tiene como motivación principal el objetivo de darle a los emisores mecanismos para verificar que los elementos de la red no eliminan el codepoint CE, y además que los receptores reportan correctamente al emisor los paquetes con el codepoint CE establecido, como es requerido por el protocolo de transporte.

El codepoint "00" (codepoint Not-ECT) indica que el paquete transportado no posee compatibilidad con ECN. El codepoint "11" (codepoint CE) es usado por el router para indicarle congestión a los nodos extremos de la conexión. Cuando un paquete arriba a un router congestionado, éste es descartado de la misma forma que se realizaría en la ausencia de ECN.

Tras la recepción de un paquete CE en la capa de transporte con soporte ECN, los algoritmos de control de congestión, como así también las respuestas de los controles de congestión, deben ser esencialmente los mismos a los utilizados ante el descarte de un paquete. Por ejemplo, para TCP con soporte a ECN, el emisor requiere reducir a la mitad su ventana de congestión para cualquier ventana de datos conteniendo ya sea un paquete descartado o una indicación ECN.

Como se describió en la sección 2.3.2, debido a la historia inestable del octeto ToS el uso del campo ECN no posee garantías de compatibilidad con los usos pasados de los dos bits que preceden a ECN.

Una razón para requerir que la respuesta del control de congestión al arribo de un paquete CE sea esencialmente la misma que la respuesta aun paquete descartado es adaptar el desarrollo incremental de ECN en los sistemas finales y en los routers. Si hubiese diferentes respuestas del control de congestión para un codepoint CE que para un paquete descartado, esto podría resultar en un tratamiento injusto para distintos flujos.

Una meta adicional es que los sistemas finales reaccionen a una congestión como máximo una vez por cada ventana de datos (es decir, como máximo una vez por cada *Round-TripTime*), para evitar reaccionar varias veces a múltiples indicaciones de congestión en un *Round-Trip Time*.

Para un router, el codepoint CE de un paquete compatible con ECN solo debería ser establecido si el router habría, de otra forma, desechado el paquete como una forma de indicación de congestión a los nodos finales. Cuando el buffer del router no está todavía completamente lleno y el router está preparado para descartar un paquete para informar a los nodos extremos posible congestión, debería primero asegurarse de ver si el codepoint ECT está presente en el encabezado IP de ese paquete. De ser así, en lugar de descartar el paquete, el router debería establecer el codepoint CE en dicho paquete.

Cuando router recibe un paquete CE, el codepoint CE no es modificado y su transmisión es efectuada de forma usual. Si se produce una congestión severa y el buffer del router está lleno, entonces el router descartará paquetes a medida que otros vayan arribando. Este tipo de pérdidas de paquetes es relativamente poco frecuente cuando la mayoría de los sistemas finales son compatibles con ECN y participan sobre TCP u otro mecanismo de control de congestión compatible. En un ambiente apropiadamente compatible con ECN, la perdida de paquetes debería ocurrir principalmente durante transiciones o en presencia de fuentes no cooperativas.

Lo mencionado anteriormente sobre cuando se podría establecer el codepoint CE en vez de descartar el paquete se aplica por defecto para todas los Per-Hop Behaviors (PHBs) en los servicios diferenciados. Además, las especificaciones para PHBs pueden proveer mayores detalles en cómo una implementación compatible elegirá entre CE y descartar el paquete, aunque no es requerido. Un router no debe establecer el CE en vez de descartar el paquete cuando las razones de descarte son distintas a las de congestión o el deseo de indicar inicio de congestión a los nodos extremos.

2.5.2. La fragmentación IP y ECN

Los paquetes compatibles con ECN pueden ser fragmentados y el re-ensamblado de estos paquetes no debe perder las notificaciones de congestión. En otras palabras, si cualquier fragmento de un paquete IP a ser re-ensamblado tiene el codepoint CE establecido, debe tomarse alguna de las siguientes acciones:

- Establecer el codepoint CE en el paquete re-ensamblado. Sin embargo, esto no debe ocurrir si alguno de los otros fragmentos que contribuyen en el re-ensamblado trae el codepoint Not-ECT.

- El paquete es descartado, en lugar de ser re-ensamblado, por cualquier otra razón.

Si las dos acciones son aplicables, se debe optar por alguna. El re-ensamblado de un paquete fragmentado no debe cambiar el codepoint ECN cuando todos los fragmentos traen el mismo codepoint.

Pueden surgir situaciones en las cuales el re-ensamblado especificado anteriormente no es lo suficientemente preciso. Por ejemplo, si existe una entidad maliciosa en el camino hacia o luego del punto de fragmentación, los fragmentos del paquete podrían acarrear una mezcla de codepoints ECT(0), ECT(1), y/o Not-ECT. La especificación de re-ensamblado anteriormente descripta no establece requerimientos para el re-ensamblado de fragmentos en este caso.

2.5.3. Soporte desde el protocolo de transporte

Además de la funcionalidad dada por el campo ECN en el encabezado del paquete IP, ECN requiere soporte desde el protocolo de transporte. Este puede requerir de una negociación entre los puntos extremos durante la configuración para determinar que todos estos son compatibles con ECN, de esta forma el emisor puede establecer el codepoint ECT en los paquetes transmitidos. Por otro lado, el protocolo de transporte debe ser capaz de reaccionar apropiadamente ante la recepción de paquetes CE. Esta reacción podría ser en forma de:

- El receptor de los datos informando al emisor de los datos el arribo del paquete CE (por ejemplo, TCP).

- El receptor de los datos retirándose de un grupo multicast en capas (ejemplo, Receiver-driven Layered Multicast).

- De alguna otra acción que finalmente reduce la taza de arribo de ese flujo en el enlace congestionado.

Los paquetes CE indican congestión persistente más que transitoria, y por lo tanto las reacciones al paquete CE recibido deberían ser apropiadas para la congestión persistente.

En TCP, ECN requiere tres nuevas funcionalidades:

1. Negociación entre los extremos durante la configuración de la conexión para determinar si ambos son compatibles con ECN.

2. Una bandera ECN-Echo (ECE) en el encabezado TCP de manera que el receptor de los datos pueda informar al emisor cuando se ha recibido un paquete CE.

3. Una bandera de Congestion Window Reduced (CWR) en el encabezado TCP de forma que el emisor de los datos pueda informar al receptor que la ventana de congestión ha sido reducida.

El soporte requerido por otros protocolos de transporte probablemente sea diferente.

Los mecanismos de TCP para negociar la compatibilidad con ECN utilizan la bandera ECN-ECHO (ECE) en el bit 9 del campo reservado del encabezado TCP. Para permitir al receptor TCP determinar cuándo dejar de usar la bandera ECN-ECHO, se introduce una segunda bandera en el encabezado TCP denominada CWR. Esta bandera es asignada al bit 8 del campo reservado, dejando así 4 bits libres (del bit 4 al 7). En la Figura 2.5.2 se muestra parte del encabezado TCP con las nuevas banderas.

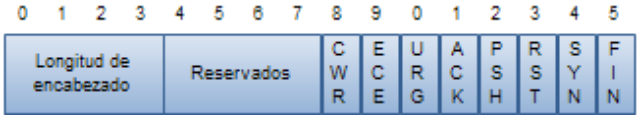


Figura 2.5.2: Definición de los bytes 13 y 14 del encabezado TCP.

ECN utiliza las banderas ECT y CE en el encabezado IP para la señalización entre los routers y los puntos extremos de la conexión, y utiliza las banderas ECN-ECHO y CWR en el encabezado TCP para la señalización entre los puntos extremos en TCP. A continuación se describe una secuencia de eventos típicos a una reacción de congestión basada en ECN para una conexión TCP y que es ilustrada por la Figura 2.5.3:

1. El codepoint ECT es establecido en los paquetes transmitidos por el emisor para indicar que ECN es soportado por las entidades de transporte de estos paquetes.

2. Un router compatible con ECN detecta congestión inminente y detecta que el codepoint ECT está establecido en el paquete que está apunto de descartar. Entonces, en vez de descartarlo, el router elije establecer el codepoint CE en el encabezado IP y reenviar el paquete.

3. El receptor recibe el paquete con el codepoint CE habilitado, y establece la bandera ECN-ECHO en su próximo TCP ACK enviado al emisor.

4. El emisor recibe el paquete con el TCP ACK con el ECN-ECHO establecido y reacciona a la congestión como si un paquete hubiese sido descartado.

5. El emisor establece la bandera CWR en el encabezado TCP en el siguiente paquete enviado al receptor, notificando el arribo del paquete con el ECN-ECHO.

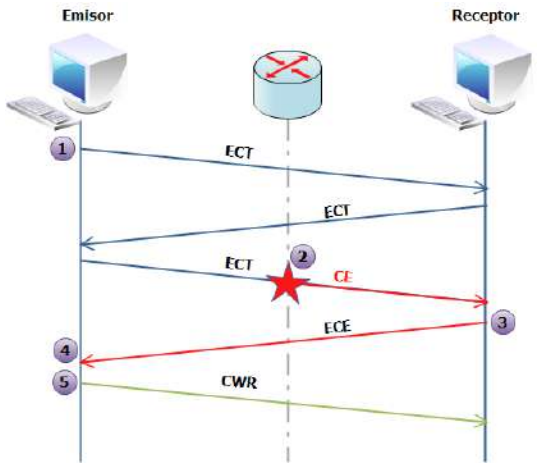


Figura 2.5.3: Secuencia de notificación de congestión.

2.5.4. La iniciación TCP

En la fase de configuración de conexión TCP, las partes intervinientes intercambian información acerca de su intención de usar ECN. Completada esta negociación, el emisor TCP establece un codepoint ECT en el encabezado IP de los paquetes de datos para indicarle a la red que las capas de transporte de los extremos soportan ECN y lo utilizarán en aquellos paquetes que contengan

dicho codepoint activado. Esto le indica a los routers intermedios de la comunicación que pueden establecer el codepoint CE en estos paquetes si desean utilizar a los mismos como un método de notificación de congestión. Si la conexión TCP no desea utilizar la notificación ECN en un paquete en particular, el emisor del paquete establece el codepoint not-ECT, y el receptor TCP ignora el codepoint CE en el paquete recibido.

Para explicar mejor este proceso se designarán nombres a los elementos intervinientes de la negociación:

- **Host A:** Iniciador de la conexión.
- **Host B:** Destino de la conexión.
- **Paquete SYN ECN-setup:** Paquete SYN con las banderas ECE y CWR establecidas.
- **Paquete SYN non-ECN-setup:** Paquete SYN con al menos una de las banderas ECE y CWR no establecidas.
- **Paquete SYN-ACK ECN-setup:** Paquete SYN-ACK con solo la bandera ECE establecida (la bandera CWR no estará establecida).
- **Paquete SYN-ACK non-ECN-setup:** Paquete SYN-ACK con cualquier otra configuración de bandera ECE y CWR.

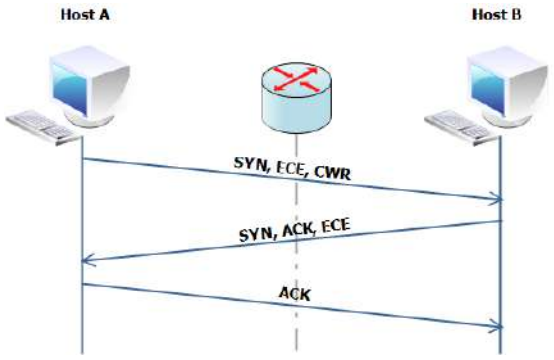


Figura 2.5.4: Establecimiento de conexión TCP de 3 vías con ECN.

Antes de que una conexión TCP pueda utilizar ECN, el Host A envía un paquete SYN ECN-setup, y al recibirlo el Host B envía un paquete SYN-ACK ECN-setup (Figura 2.5.4). Para un paquete SYN, el establecimiento de las banderas ECE y CWR al mismo tiempo en el paquete SYN ECN-setup está definido como una indicación de que el emisor es compatible con ECN, más que una indicación de

congestión o una respuesta a la misma. Más precisamente, un paquete SYN ECN-setup indica que la implementación TCP que transmite el paquete SYN participará en una conexión ECN tanto el emisor como el receptor. En otras palabras, tanto el Host A como el Host B, al recibir paquetes de datos con el codepoint CE establecido en el encabezado IP responderán estableciendo la bandera ECE en los paquetes de respuesta TCP (ACKs). Al recibir dichos paquetes responderán, cuando sea apropiado, mediante la reducción de su ventana de congestión y estableciendo la bandera CWR en su paquete de respuesta. Un paquete SYN ECN-Setup no hace que el emisor TCP tenga que establecer el codepoint ECT en alguno o todos los paquetes que este transmita. Sin embargo, el hecho de responder apropiadamente a los paquetes con el codepoint CE permanece aun cuando el emisor TCP envía un paquete SYN sin establecer los bits ECE y CWR dentro de esta conexión TCP.

Cuando el Host B envía un paquete SYN-ACK ECN-Setup, envía la bandera ECE establecida pero no la CWR. Un paquete SYN-ACK ECN-Setup está definido como una notificación que el host TCP transmitiendo el SYN-ACK es compatible con ECN. Como sucedía con el paquete SYN, un paquete SYN-ACK ECN-Setup no obliga al host TCP establecer el codepoint ECT en los paquetes transmitidos.

Las siguientes reglas se aplican al envío de paquetes ECN-Setup dentro de una conexión TCP cuando ésta es definida por las reglas estándares de establecimiento y terminación de una conexión TCP.

- Si un host recibe un paquete SYN ECN-Setup, entonces puede, si lo desea, enviar un paquete SYN-ACK ECN-Setup. De otra forma, si no recibe un paquete SYN ECN-Setup, no debe enviar un paquete SYN-ACK ECN-Setup.

- Un host no debe establecer el codepoint ECT en paquetes de datos a menos que este haya enviado por lo menos un paquete SYN ECN-Setup o SYN-ACK ECN-Setup, y ha recibido por lo menos un paquete SYN ECN-Setup o SYN-ACK ECN-Setup, y no ha enviado paquetes paquete SYN non-ECN-Setup o SYN-ACK non-ECN-Setup. Si un host ha recibido por lo menos un paquete SYN non-ECN-Setup o SYN-ACK non-ECN-Setup, entonces no debería establecer ECT en los paquetes de datos.

- Si un host alguna vez estableció el codepoint ECT en un paquete de datos, entonces, ese host debe establecer/eliminar correctamente el bit CWR de TCP en todos los paquetes subsecuentes en la conexión.

- Si un host ha enviado por lo menos un paquete SYN ECN-Setup o SYN-ACK ECN-Setup, y no ha recibido paquetes SYN non-ECN-Setup o SYN-ACK non-ECN-Setup, entonces al recibir paquetes de datos TCP con los codepoints ECT y CE establecidos en el encabezado IP deberá procesarlos como se especificó para una conexión compatible con ECN.

- Un host que no está dispuesto a usar ECN sobre una conexión TCP debería borrar ambas banderas ECE y CWR en todos los paquetes SYN non-ECN-Setup y/o SYN-ACK non-ECN-Setup que este envía. Los receptores deben manejar correctamente todas las formas de paquetes SYN non-ECN-Setup y SYN-ACK non-ECN-Setup.

- Un host no debe establecer ECT en paquetes SYN o SYN-ACK.

Un cliente TCP entra en estado TIME-WAIT después de recibir un FIN-ACK, y cambia al estado CLOSED después de un tiempo de espera (time-out). Muchas implementaciones TCP crean una nueva conexión TCP si reciben un paquete SYN dentro de la ventana de tiempo durante el estado TIME-WAIT. Cuando un host TCP entra en estado TIME-OUT o CLOSED, debería ignorar cualquier estado anterior acerca de la negociación de ECN para esa conexión.

Existe una cuestión por el cual se elige tener que enviar el Paquete SYN ECN-setup con las banderas ECE y CWR establecidas, mientras que la respuesta a este paquete (es decir, el Paquete SYN-ACK ECN-setup) solo establece la bandera ECE. Esta asimetría es necesaria para que la negociación de la capacidad ECN sea robusta para algunas implementaciones TCP. Existe por lo menos una implementación defectuosa de TCP en la cual los receptores TCP establecen el campo reservado del encabezado TCP en los paquetes ACK (y por lo tanto, los SYN-ACK) reflejando el campo reservado del encabezado TCP de los paquetes de datos recibidos. Dado que el paquete TCP SYN establece las banderas ECE y CWR para indicar compatibilidad ECN, mientras que los

paquetes SYN-ACK solo establecen la bandera ECE, el emisor TCP interpreta correctamente que el receptor no es compatible con ECN si encuentra un reflejo de sus propias banderas. De esta forma, el emisor TCP no es engañado por la interpretación defectuosa del TCP enviando un paquete SYN-ACK que simplemente refleja el campo reservado del paquete SYN entrante. Como puede observarse, bajo esta implementación de TCP no se puede trabajar con ECN.

2.5.5. Emisor TCP

En una conexión TCP compatible con ECN, los paquetes de datos son transmitidos con un codepoint ECT en el encabezado IP. Si es necesario indicar compatibilidad con notificación de congestión explícita con solo uno de los codepoints ECT, este debería ser el ECT(0). Si el emisor recibe un paquete ACK ECE (un paquete ACK con la bandera ECE), este sabrá que existe una congestión en la red de la comunicación. La notificación de congestión debe ser tratada solo como una pérdida por congestión como en una conexión TCP sin soporte ECN. Esto es, el emisor TCP divide a la mitad su ventana de congestión “cwnd” (cantidad máxima de bytes que puede transmitir sin esperar la llegada de un ACK) y reduce el umbral slow-start “ssthresh”, el cual indica el tamaño aproximado de la ventana de congestión del receptor. El emisor TCP nunca debería incrementar la ventana de congestión en repuesta a la recepción de un paquete ACK ECE.

TCP no debería reaccionar a las indicaciones de congestión más de una vez por ventana de datos. Esto es, la ventana de congestión del emisor TCP debería ser reducida solo una vez en respuesta a una serie de pérdidas de paquetes y/o a una serie de paquetes CE desde una misma ventana de datos. Adicionalmente, el emisor TCP no debería disminuir el umbral ssthresh, si ya ha sido reducido dentro del último round-trip time. Sin embargo, si cualquier paquete retransmitido es descartado, entonces esto es interpretado por el emisor TCP como una nueva instancia de congestión.

Luego que el emisor TCP redujo su ventana de congestión en respuesta a un paquete CE, los ACKs que continúan arribando permiten enviar más paquetes hasta tanto la ventana de congestión lo admita. Si la ventana de congestión consiste de un solo MSS (segmento de tamaño máximo), y el emisor TCP recibe un paquete ACK ECN-ECHO, entonces, en principio aún reduce a la mitad su ventana de congestión. Sin embargo, el valor de la ventana de congestión está limitado a un MSS como valor mínimo. Si el emisor TCP continua transmitiendo, usando una ventana de congestión

de un MSS, provocará la transmisión de un único paquete por round-trip time. Si se recibe un nuevo ECN-ECHO y la ventana de congestión es de un MSS, entonces habrá que utilizar un temporizador de transmisión como un mecanismo de reducir más la tasa, el cual permitirá enviar un paquete cada vez que este caduque. Además, el emisor TCP deberá reiniciar el temporizador de trasmisión cuando recibe un paquete ECN. De esta forma, el emisor TCP estará habilitado a enviar un nuevo paquete sólo cuando el tiempo de retransmisión expire.

Cuando un emisor TCP con soporte ECN reduce su ventana de congestión por cualquier razón, establecerá la bandera CWR en el encabezado TCP en el primer paquete de datos que envíe luego de haberla reducido. Si ese paquete de datos es descartado en la red, entonces el emisor TCP tendrá que reducir nuevamente la ventana de congestión y retransmitir el paquete descartado. De esta forma, el bit CWR en el encabezado TCP no debería ser establecido en paquetes retransmitidos.

Cuando el emisor de datos TCP está listo para establecer el bit CWR después de haber reducido su ventana de congestión, este debería hacerlo solo en el primer paquete de datos nuevo a transmitir.

2.5.6. Receptor TCP

Cuando el receptor TCP recibe un paquete de datos con el codepoint CE, establecerá la bandera ECN-ECHO en el encabezado TCP en los paquetes ACKs subsecuentes. Para cualquier implementación TCP “delayed-ACK”, en las cuales el receptor TCP envía un ACK por cierta cantidad de paquetes de datos recibidos (en lugar de enviar un ACK por cada paquete recibido), los paquetes ACKs subsecuentes deberán establecer la bandera ECN-ECHO en “1” solo si alguno de los paquetes de datos recibidos para dicho ACK posee el codepoint CE establecido. Esta repetición de notificación se realiza para proveer robustez frente a la posibilidad del descarte de un paquete ACK que lleva la bandera ECN-ECHO. El receptor TCP utiliza la bandera CWR recibida desde el emisor TCP para determinar cuándo dejar de enviar la bandera ECN-ECHO en los paquetes.

Luego de recibir un paquete con la bandera CWR, los ACKs subsecuentes para paquetes de datos non-CE no poseerán la bandera ECN-ECHO establecida. Si otro paquete CE es recibido por el receptor, este nuevamente enviaría paquetes ACKs con la bandera ECN-ECHO establecida hasta

que la recepción de un paquete CWR vuelva a garantizar que el emisor redujo su ventana de congestión aún más.

2.5.7. Retransmisión de paquetes TCP

Las implementaciones TCP compatibles con ECN no deben establecer ninguno de los codepoint ECT (ECT(0) o ECT(1)) en la cabecera IP para los paquetes de datos retransmitidos, y el receptor de datos TCP debe ignorar el campo ECN de todos los paquetes que arriban fuera de su ventana de tiempo actual. Esto se realiza para una mayor seguridad frente a ataques de denegación de servicio, como así también para la robustez de la notificación ECN con paquetes que son descartados luego en la red.

Es fácil notar que si un extremo TCP establece un codepoint ECT en un paquete retransmitido, entonces el otro extremo nunca recibirá el codepoint CE establecido por el router congestionado en el camino de la comunicación. Esto hace que la notificación congestión no sea robusta, incluso para paquetes que luego son descartados en la red.

Adicionalmente, un atacante capaz de realizar una sustitución de dirección IP fuente del emisor TCP (lo que se conoce como spoofing) podría enviar paquetes de datos con números de secuencia arbitrarios y con el codepoint CE establecido en el encabezado IP. En la recepción de este paquete adulterado, el receptor de datos TCP podría determinar que los datos no se encuentran en la ventana de recepción actual, y devolver un ACK duplicado. Se llamará paquete fuera-de-ventana en el receptor de datos TCP al paquete de datos que se encuentra fuera de la ventana actual del receptor. En la recepción de un paquete de este tipo, el receptor de datos TCP debe decidir cuándo o no debe tratar el codepoint CE como una notificación válida de congestión, y por lo tanto cuándo devolver una notificación ECN-ECHO al emisor TCP. Si el receptor de datos TCP ignora el codepoint CE en un paquete *fuera-de-ventana*, entonces el emisor TCP no recibiría esa posible y legítima notificación de congestión de la red, resultando en una violación al control de congestión extremo a extremo. Por otro lado, si el receptor de datos TCP acepta la indicación CE del paquete *fuera-de-ventana*, y reporta la notificación de congestión al receptor de datos TCP, entonces el nodo malicioso que creó el paquete adulterado logró atacar satisfactoriamente la conexión TCP forzando al emisor de datos a reducir su ventana de congestión innecesariamente. Para prevenir tal ataque de denegación de servicios, se especifica que un emisor de datos TCP legítimo no debe

establecer un codepoint ECT en paquetes de datos retransmitidos, y que el receptor de datos TCP debería ignorar el codepoint CE de paquetes *fuera-de-ventana*.

Una desventaja de no establecer el ECT(0) o ECT(1) en paquetes retransmitidos es que deniega lo protección ECN para dichos paquetes. Sin embargo, para una conexión TCP en un ambiente completamente compatible con ECN con congestión leve, los paquetes raramente deberían ser descartados en primer lugar, y por lo tanto rara vez deberían surgir instancias de paquetes retransmitidos. Si los paquetes están siendo retransmitidos, entonces hay paquetes que ya han sido perdidos y ECN no pudo prevenir.

Si un router establece el codepoint CE para un paquete de datos compatible con ECN en una conexión TCP, entonces se garantiza que esa conexión TCP recibirá ésta notificación de congestión, o alguna otra notificación de congestión, dentro de la misma ventana de datos aún si el paquete es descartado o reordenado en la red. Se deben considerar dos casos: cuando el paquete de datos es retransmitido posteriormente, y cuando el paquete no es retransmitido posteriormente.

En el primer caso, dado que el emisor de datos ya ha retransmitido este paquete, se sabe que el emisor de datos ha respondido a una notificación de congestión para algún paquete dentro de la misma ventana de datos que el paquete original. Por lo tanto, aún si la primera transmisión del paquete es descartada en la red, o retrasada, si éste tiene el codepoint CE establecido, y es luego ignorado por el receptor de datos como un paquete *fuera-de-ventana*, no es un problema dado que el emisor ya ha respondido a una notificación de congestión para esa ventana de datos.

En el segundo caso, si el paquete jamás es retransmitido por el emisor de datos, entonces este paquete de datos es la única copia de estos datos recibidos por el receptor, y por lo tanto arriban al receptor como un paquete dentro de la ventana, sin importarle cuán retrasado o reordenado llega el paquete. En este caso, si el codepoint CE está establecido en el paquete dentro de la red, será tratado por el receptor de datos como una notificación de congestión válida.

2.5.8. Incompatibilidad entre los nodos extremos

Aún en ambientes incompatibles con ECN, existen preocupaciones serias acerca del daño que puede ser causado por flujos no compatibles o insensibles (esto es, flujos que no responden a

notificaciones de control de congestión reduciendo su tasa de arribo al enlace congestionado). Por ejemplo, un nodo final puede deshabilitar el control de congestión no reduciendo su ventana de congestión en respuesta de un paquete descartado. Esto es una preocupación para la Internet actual, por lo cual especialistas en el tema han argumentado que se deberán desarrollar mecanismos para que los routers puedan detectar y tratar de forma diferenciada a paquetes de flujos no compatibles. También, se ha sugerido que técnicas tales como planificación extremo a extremo por flujo y aislamiento de un flujo de otro, servicio diferenciado, o reservaciones extremo a extremo pueden eliminar algunos de los efectos más dañinos de los flujos insensibles.

Puede parecer que el descarte de paquetes en sí mismo es un impedimento adecuado para la incompatibilidad, y que el uso de ECN lo elimina. Se podría decir en respuesta a esto que en routers compatibles con ECN el comportamiento de descarte de paquetes se mantiene en momentos de alta congestión e incluso en tiempos de alta congestión, el descarte de paquetes no es por sí mismo un impedimento adecuado para la incompatibilidad.

Por un lado, los routers compatibles con ECN podrán solamente marcar paquetes (en lugar de descartarlos) cuando la tasa de marcado de paquetes es razonablemente baja. Durante los periodos donde el promedio de tamaño de la cola excede el umbral superior, y así la potencial tasa de marcado de paquetes podría ser alto, la recomendación es que los routers descarten paquetes en vez de establecer el codepoint CE en los encabezados de los paquetes.

Por otro lado, cuando puede desarrollarse ECN durante los períodos de tasa de marcado de paquetes baja o moderada, podría producirse un pequeño efecto disuasivo en flujos insensibles de descarte más que de marcado de esos paquetes. Por ejemplo, flujos con retraso intensivo utilizando una entrega confiable pueden tener un incentivo de incrementar más que disminuir su tasa de envió en presencia de paquetes descartados. De la misma forma, flujos con retraso sensible usando una entrega no confiable puede incrementar el uso de FEC (Forward error correction) en respuesta de incremento en la tasa de paquetes descartados, aumentando más que disminuyendo su tasa de envío. Por estas mismas razones, no se cree que el descarte de paquetes por sí mismo sea un impedimento efectivo para la incompatibilidad, incluso en un ambiente de tasas altas de descarte de paquetes cuando todos los flujos están compartiendo la misma tasa de

paquetes descartados.

Se han propuesto varios métodos para identificar y restringir flujos incompatibles o insensibles. La implementación de ECN al ambiente de una red no podrá, en ninguna forma, incrementar la dificultad del diseño y despliegue de tales mecanismos. En todo caso, la incorporación de ECN a la arquitectura haría levemente más fácil el trabajo de identificar flujos insensibles. Por ejemplo, en un ambiente compatible con ECN, los routers no están limitados a la información acerca de paquetes que son descartados o tienen el codepoint CE establecido a ese router especifico; en tal ambiente, los routers también podrían tomar nota de paquetes CE que arriban indicando congestión en algún lugar anterior del camino recorrido por dicho paquete.

2.5.9. Robo y Denegación de servicio

La correcta operación de ECN requiere de la cooperación del receptor para devolver las señales CE al emisor, pero el protocolo carece de mecanismos que obliguen esta cooperación. Esto da la posibilidad de que receptores malintencionados o mal implementados puedan ignorar los codepoints CE establecidos por un router que detecta congestión en el camino de los datos transportados. De esta forma, el nodo estaría obteniendo un beneficio por encima de los demás ante un evento de congestión al no indicarle al emisor de los datos que debe reducir su ventana de congestión, mientras el resto de los flujos que transitan por el mismo camino realizan esta reducción (Figura 2.5.5). Un router también podría estar trabajando en forma maliciosa, modificando cualquiera de los bits del campo ECN. Esta modificación podría ser detectada en IPv4 por el campo checksum del encabezado IP, pero no es posible en el caso de IPv6 ya que no existe tal control. La modificación de los bits del campo ECN podría consistir en reportes de congestión falsos, deshabilitar compatibilidad con ECN para un paquete individual, borrar la notificación de congestión, o indicar falsamente compatibilidad ECN. Esto podría terminar en la falla del control de congestión para un flujo, dando como resultado un incremento de la congestión de la red, y resultando finalmente en descartes de paquetes subsecuentes para este flujo, denegando un correcto tratamiento.

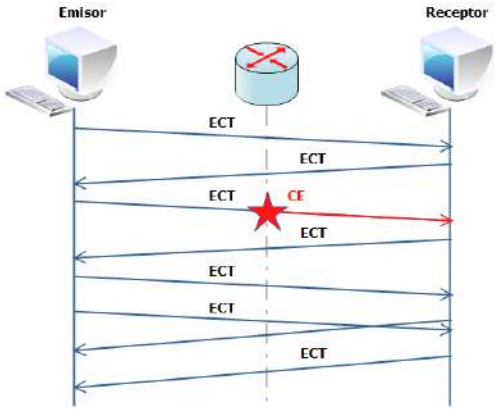


Figura 2.5.5: Robo de servicio en ECN.

Estos problemas pueden ser remediados mediante el uso de un bit Nonce (Figura 2.5.6), el cual fue propuesto en el RFC 3540 (N. Spring, 2003) aunque éste todavía se encuentra como un documento experimental y no un estándar. El ECN Nonce le permite al emisor verificar el correcto comportamiento del receptor y que no existe ninguna otra interferencia que oculte paquetes marcados (o descartados) en el camino. El ECN Nonce la infraestructura ECN frente a errores de implementación o de abuso deliberado.

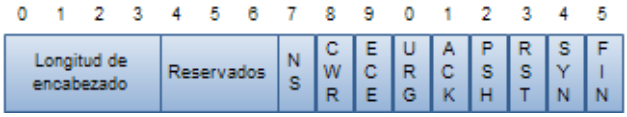


Figura 2.5.6: Definición en el RFC 3540 del bit Nonce Sum en el encabezado TCP.

El uso de ECN Nonce permite:

- Detectar con alta probabilidad a un receptor malintencionado, pero nunca acusar a un receptor que presenta un comportamiento correcto. Es decir, no emite falsos positivos.
- No modificar otros aspectos de ECN ni reducir sus beneficios.
- Disminuir tanto el overhead por paquetes como los requerimientos de procesamiento, dado que se trata solo de una bandera en el encabezado TCP.
- Simplicidad y no es propenso a otros ataques.

El protocolo ECN para TCP permanece sin cambios, salvo por la definición de un nuevo campo en el encabezado TCP. Como se definió anteriormente, los ECT(0) y ECT(1) (ECN-Capable Transport) son establecidos en el campo ECN del encabezado IP en los paquetes de salida. Los routers congestionados cambian este campo al valor CE. Cuando los receptores detectan este valor, establecen la bandera ECE en todos los paquetes ACKs subsecuentes hasta que reciben un paquete con la bandera CWR. Esta bandera es enviada para indicar que el emisor reaccionó a la congestión.

Un valor de bit aleatorio (Nonce) es codificado en los dos codepoints ECT. Además de estos valores, se agrega un bit suma de estos Nonces en las banderas del encabezado TCP, denominado Nonce sum (NS). El marcado de paquetes elimina el valor Nonce en el codepoint ECT, dado que CE sobrescribe los dos bits ECN en el encabezado IP. Dado que cada Nonce es requerido para calcular la suma, ésta solo será correcta para la recepción de paquetes sin marcar. El bit Nonce no solo previene a los receptores de cancelar a los paquetes marcados, sino que también los otros elementos en el camino de la red no pueden desmarcar paquetes satisfactoriamente acertando al valor Nonce original.

El emisor puede verificar la suma Nonce retornada por el receptor para asegurarse que la notificación de congestión, en forma de paquetes marcados o descartados, no está siendo ocultada. Dado a que la suma Nonce es solamente de un bit de longitud, el emisor tiene 50% de posibilidades de detectar un receptor mentiroso. Ya que cada ACK es independiente, los tramposos serán detectados rápidamente si ellos repiten las señales de congestión.

La comunicación TCP empezará con una suma Nonce inicial a la cual se le sumará, a través de un or-exclusivo o una suma de paridad, el valor de ECT elegido para el primer paquete y se establecerá dicho valor en el bit NS del encabezado TCP. El ECT será seleccionado aleatoriamente entre el ECT(0) y el ECT(1). El receptor emitirá el ACK correspondiente manteniendo el mismo valor de NS. En la Figura 2.5.7 se muestra la forma en que se establece la suma Nonce y como es verificada por el emisor.

Después que ha ocurrido congestión y los paquetes han sido marcados o perdidos, es necesaria una re-sincronización de las sumas nonces entre el receptor y el emisor. Cuando los paquetes son

marcados el nonce es borrado, y la suma de los nonces en el receptor no coincidirá más con la suma en el emisor y esta diferencia será constante. Es posible re-sincronizar el emisor y el receptor luego de la congestión a través del establecimiento de la suma nonce al valor de la suma del receptor. Debido a que la notificación de congestión no necesita ser transmitida más de una vez por Round Trip time, el emisor suspende la validación de la suma mientras la señal CWR está siendo transmitida y re-establece la suma cuando recibe el ACK correspondiente a dicha señal. Esto tiene el beneficio de que el receptor no está explícitamente involucrado en el proceso de re-sincronización. Este proceso es mostrado en la Figura 2.5.8.

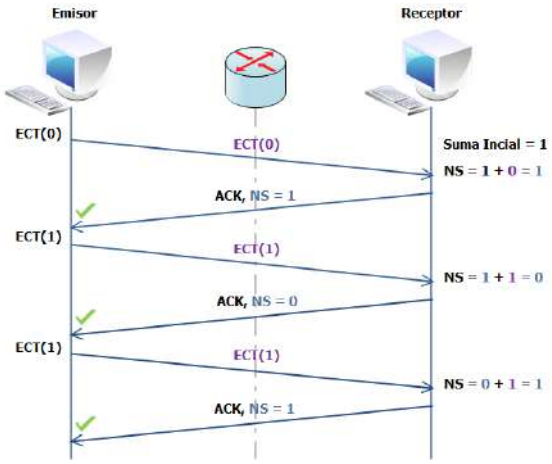


Figura 2.5.7: Suma de bits Nonce en el receptor.

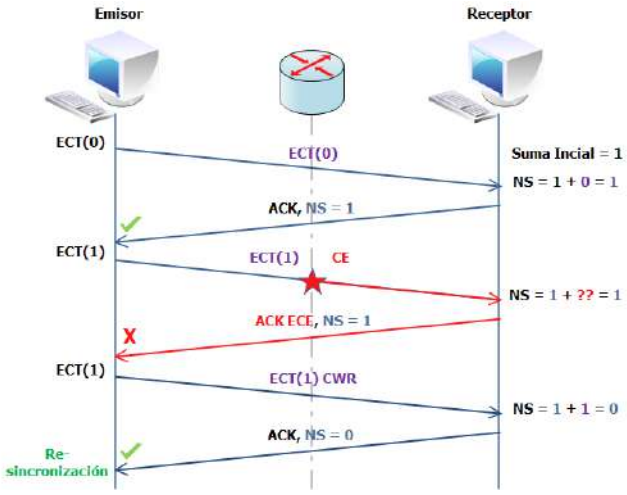


Figura 2.5.8: Suma de bits Nonce y re-sincronización de sumas.

2.6. IPsec

IPsec es un conjunto de protocolos que brindan servicios de seguridad para conexiones de red definido y detallado por el RFC2401 (S. Kent, 1998). El estudio e implementación de IPsec puede volverse complejo dado a que éste provee mecanismos y no políticas, para ser más concisos, IPsec no define un algoritmo de encriptación o una función específica de autentificación, sino que provee un framework que le permite a una implementación proveer casi cualquier cosa que dos extremos hayan acordado. Queda evidente entonces, que el grado de efectividad o de seguridad que IPsec brinda se encuentra estrechamente dependiente de los algoritmos que se empleen y de su correcta implementación. Al ser éste un campo muy amplio, solo se explicará sin entrar en detalles profundos, ni conceptos complejos relativos a seguridad, los componentes principales de IPsec necesarios para comprender exactamente cuál es su función y cómo trabaja.

Los objetivos para los cuales IPsec está diseñada son interoperabilidad, alta calidad, seguridad basada en criptografía para IPv4 e IPv6. Este conjunto de servicios de seguridad ofrecidos incluyen: control de acceso, integridad sin conexión, autentificación de origen de datos, protección contra repeticiones, confidencialidad (por medio de encriptación). Estos servicios son provistos en la capa IP, ofreciendo protección para IP y/o protocolos de capas superiores.

Estos objetivos son alcanzados por medio de la implementación de dos protocolos de seguridad de tráfico, el encabezado de autentificación (AH), y el encapsulamiento de payload (ESP), los cuales requerirán el uso de procedimientos y protocolos de administración de claves criptográficas. Ambos protocolos pueden ser aplicados individualmente o combinarse para proveer un conjunto deseado de servicios de seguridad en IPv4 e IPv6. Cada protocolo soporta dos modos de uso: modo de transporte, provee protección para los protocolos de capas superiores; y modo de túnel, provee protección a paquetes IP dentro de túneles IP.

El conjunto de protocolos IPsec pueden ser empleados en cualquier contexto y la manera en que serán empleadas dependerá por los requerimientos de seguridad y los requerimientos del sistema de usuarios, aplicaciones, y/o los sitios de las organizaciones.

Cuando estos mecanismos son correctamente implementados y desarrollados, IPsec no debe afectar a los usuarios, hosts, u otros componentes de Internet que no empleen estos mecanismos

de seguridad para la protección de su tráfico. Estos mecanismos además, fueron desarrollados para ser algoritmos independientes. Esta modularidad permite la selección de diferentes conjuntos de algoritmos sin afectar las otras partes de la implementación y brindar servicios de seguridad totalmente a medida.

Como se mencionó anteriormente, IPsec provee seguridad en la capa IP permitiendo a los sistemas seleccionar los protocolos de seguridad requeridos, determinar los algoritmos a utilizar por los servicios, y administrar cualquier clave criptográfica requerida para proveer los servicios peticionados. IPsec puede ser implementado tanto para proteger uno o más caminos entre un par de hosts, un par de Gateways seguros, o entre un Gateway seguro y un host. El término "Gateway seguro" o "Gs" es utilizado para hacer referencia a un sistema intermedio que implementa protocolos IPsec. Por ejemplo, un router o un firewall implementando IPsec son considerados un Gateway seguro.

2.6.1. Encabezado de autenticación (Authentication Header, o AH)

El encabezado de autentificación (AH) es utilizado para autenticar el tráfico IP y puede cumplir un triple propósito: asegurarnos que estamos hablando con quien pensamos que estamos hablando (y no un impostor), detectar alteraciones de los datos durante su transmisión, y opcionalmente para defendernos de réplicas realizadas por atacantes, los cuales capturan datos de la red para luego intentar re-inyectarlos a la red fuera de tiempo. No obstante, AH no provee encriptación con lo cual, si se desea confidencialidad este protocolo no la provee.

La autenticación es realizada mediante el cálculo de un hash criptográfico basado en un código de autenticación de mensaje (conocido como Hashed Message Authentication Code o HMAC) el cual toma en cuenta casi la totalidad de los campos del paquete IP (con exclusión de los que podrían ser modificados en la transmisión, tal como el TTL o el checksum), y lo almacena en el encabezado AH agregado al paquete y enviado al otro extremo.

La Figura 2.6.1 ilustra al encabezado AH, el cual contiene solamente cinco campos, ubicado entre el encabezado IP original y el payload. Estos campos son:

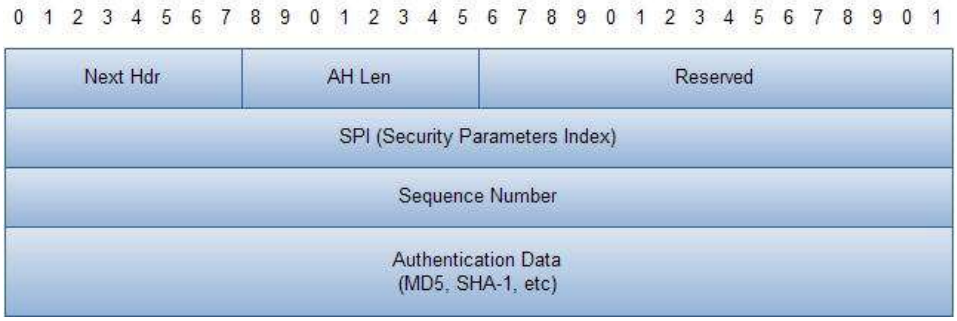


Figura 2.6.1: Encabezado AH.

- **Próximo encabezado (Next Hdr):** Identifica al protocolo al que debe entregarse el payload, y es el protocolo original del paquete antes de ser encapsulado. Los encabezados IPsec son enlazados mediante este campo.

- **Longitud AH (AH Len):** Este campo define la longitud, en palabras de 32 bits, del encabezado AH completo.

- **Reservado (Reserved):** Este campo está reservado para uso futuro y debe encontrarse relleno de ceros.

- **Índice de parámetros de seguridad (Security Parameters Index o SPI):** Este campo de 32 bits es un identificador "opaco" que ayuda al receptor del paquete a seleccionar la conversación, de todas las conversaciones en curso, a la que el paquete pertenece. Cada conexión protegida por AH, implica un algoritmo hash (MD5, SHA-1, etc.), algún tipo de información secreta y una serie de otros parámetros. El SPI es un índice para enlazar fácilmente al paquete con dichos parámetros de seguridad.

- **Número de secuencia (Sequence Number):** Identificador auto-incremental utilizado para aplicar protección ante repeticiones. Este valor está incluido dentro de los datos de autenticación, de esta manera puede detectarse cualquier modificación intencional o accidental.

- **Datos de autenticación (Authentication Data):** Este es un valor de comprobación de integridad (Integrity Check Value). El receptor del paquete recalcula el mismo hash, en caso de calcular un valor distinto al de este campo, se descarta al paquete. Esto puede ocurrir en paquetes dañados durante la transmisión o por la posesión de una clave secreta incorrecta.

2.6.1.1. AH en Modo de Transporte

Tanto AH como ESP trabajan en dos modos, modo transporte y modo túnel. El más fácil de comprender es el modo transporte el cual utilizado para proteger una conversación extremo a extremo entre dos hosts.

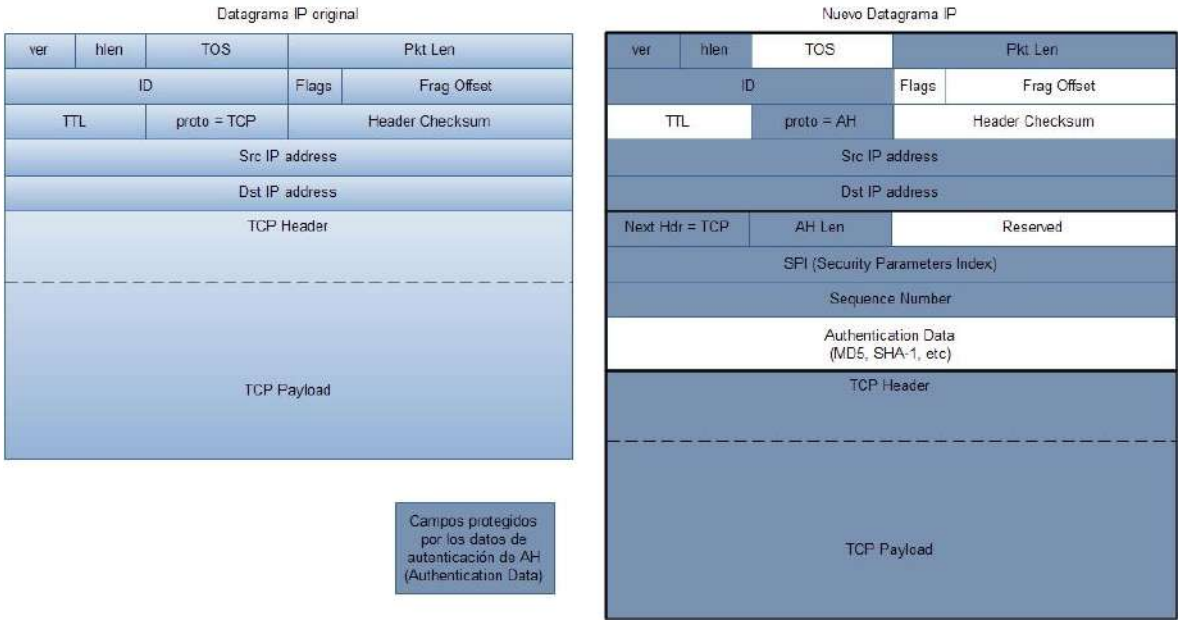


Figura 2.6.2: IPsec en AH modo transporte.

El protocolo AH en modo transporte, ilustrado en la Figura 2.6.2, el paquete IP es apenas modificado para incluir el nuevo encabezado AH entre el encabezado IP y el payload del protocolo (TCP, UDP, etc.), y el campo proto del encabezado IP es modificado, copiando el contenido original de éste al campo Next Hdr de AH, indicando que el siguiente encabezado pertenece a AH. Ésta modificación del paquete es requerida para permitir que el paquete IP original pueda ser reconstruido en el otro extremo. Luego de realizarse la validación del encabezado Ipsec al arribar en el receptor y se aprueba la comprobación de autentificación, el encabezado AH es removido y

el campo proto es restaurado a su valor original. De esta manera, el datagrama IP vuelve a su estado original y puede ser entregado al proceso en espera de éste.

2.6.1.2. AH en Modo de Túnel

El modo túnel, ilustrado en la Figura 2.6.3, posee una funcionalidad muy similar a la de una VPN, donde los paquetes IP son encapsulados dentro de otros y entregados a su destino. Al igual que el modo transporte, el paquete es sellado con un valor de comprobación de integridad para autenticar al emisor y prevenir modificaciones durante su transmisión. Sin embargo, a diferencia del modo transporte, el modo túnel encapsula tanto el encabezado IP completo como el payload. Esto permite que la dirección fuente y destino sean diferentes a las contenidas en el paquete encapsulado, haciendo posible así la creación de un túnel.

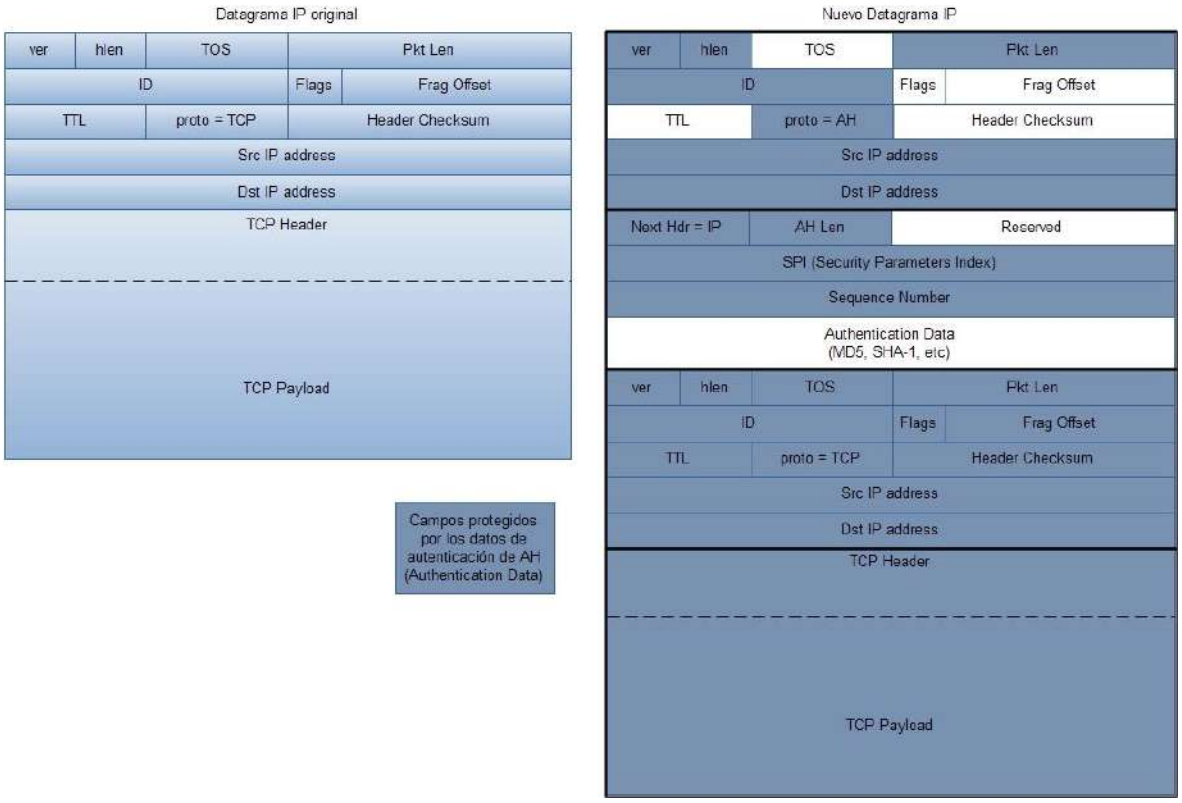


Figura 2.6.3: IPsec en AH modo túnel.

Cuando un paquete arriba a su destino por medio del modo túnel, éste es tratado por la misma comprobación de integridad como cualquier otro paquete del tipo AH. A aquellos paquetes que

superen el control mencionado, se les remueve el paquete encapsulado y el encabezado AH. De esta forma, es posible reconstruir el datagrama IP original, el cual es inyectado al proceso de enrutamiento normal. La mayoría de las implementaciones de IPsec tratan al extremo final del modo túnel como a una interface de red virtual, como lo es la interface localhost, y el tráfico entrante o saliente es sujeto a todas las decisiones de enrutamiento comunes.

El paquete reconstruido puede ser entregado a la máquina local o enviado hacia cualquier lugar según la dirección destino IP encontrado en el paquete encapsulado. A este punto, y en cualquiera de los casos, el paquete ya no posee ninguna protección de IPsec, es simplemente un datagrama IP regular. Debido a que el modo transporte es utilizado estrictamente para asegurar una conexión extremo a extremo entre dos computadoras, es normalmente implementado gateways para proveer una red virtual privada (VPN).

2.6.1.3. Consideraciones para AH (Modo transporte / Modo túnel)

Si observamos al encabezado AH y comparamos el paquete IPsec resultante en modo transporte con el de modo túnel podemos advertir que no hay ningún campo explicito que nos indique el modo de transmisión utilizado. Sin embargo, si nos ponemos a analizar las diferencias entre estos podemos darnos cuenta que la distinción del modo de transmisión es realizada mediante la inspección del campo Next Hdr del encabezado AH.

Cuando el valor del campo Next Hdr es IP, significa que el paquete en cuestión encapsula a otro paquete IP, es decir, se está utilizando el modo túnel. Cuando el campo posee cualquier otro valor (TCP, UDP, ICMP, etc.), significa que se está utilizando el modo transporte y se está asegurando a los puntos finales (host) de la conexión.

Tanto en el modo túnel como en modo transporte, el nivel superior del datagrama IP es estructurado de la misma manera, y los routers intermedios tratan a todos los tráficos IPsec de forma idéntica sin la necesidad de una inspección profunda del paquete. Cabe destacar que los host, a diferencia de los gatways, deben soportar tanto el modo transporte como el modo túnel. En cuanto a los gateways, solo necesitan soportar el modo túnel, ya que soportar el modo transporte solo es útil si se crea un punto final al mismo Gateway, tal como ocurre en las funciones de administración de red.

Otra consideración importante que se debe mantener en mente, es que IPsec en AH es incompatible con NAT (Network Address Translation). NAT es utilizado para traducir el rango de direcciones privadas a un conjunto de direcciones públicas. En este proceso, el encabezado IP es modificado durante la transmisión por el dispositivo de NAT para cambiar las direcciones IP fuente y/o destino.

Cuando se modifica una dirección IP, se obliga a recalcular el Checksum del encabezado IP. Además, este cálculo debe realizarse te todas maneras ya que el dispositivo NAT es un salto más en el camino entre el origen y el destino, lo que requiere un decremento del campo TTL. Debido a que los campos TTL y Checksum del encabezado IP son siempre modificados a lo largo del trayecto, AH los excluye de la protección. Sin embargo, no hace esto con las direcciones IP, las cuales están protegidas por el valor de comprobación de integridad (Integrity Check Value o ICV), y cualquier modificación provoca una falla al ser recibido. Como el ICV incorpora una clave secreta desconocida por las entidades intermedias, el dispositivo NAT no es capaz de recalcularlo correctamente para realizar las modificaciones pertinentes.

2.6.2. Encapsulamiento de Seguridad de Payload (Encapsulating Security Payload, o ESP)

ESP provee encriptación y autentificación (opcional), además, proporciona los modos de transporte y de túnel, los cuales trabajan de forma muy similar a lo explicado anteriormente. La capacidad de encriptar hace a ESP más complicado, ya que encapsula al payload en vez de precederlo como ocurre en AH.

Como se había mencionado, los RFCs de IPsec no estipulan el uso de ningún algoritmo de encriptación en particular, pero los más utilizados en la práctica son DES, Triple-DES, AES y Blowfish. El algoritmo que se utilizará para una conexión en particular está determinado por la Asociación de Seguridad (Security Association o SA), la cual además incluye la clave a utilizar. Las SAs son un concepto muy importante en IPsec, son utilizadas tanto por AH y ESP y serán explicadas más adelante.

ESP envuelve al payload que éste protege, tal como se muestra en la Figura 2.6.4, mediante un encabezado ESP, conformado por dos campos, y una cola ESP (conocida como ESP Trailer), conformada por 3 o 4 campos. Los campos del encabezado ESP son el Índice de parámetros de seguridad y el Número de secuencia, cuyos propósitos son idénticos que en AH. Los campos del ESP Trailer son el campo Padding (relleno), Pad Len (longitud de relleno), Next Header (tipo del próximo encabezado) y datos de autentificación. El campo Padding es provisto para permitir el uso de algoritmos de encriptación orientados a bloques, y el tamaño de estos bloques está representado por el campo Pad Len.

La autentificación opcional que provee ESP posee el mismo HMAC explicado en AH. Sin embargo, esta autentificación solo se realiza sobre el encabezado ESP y el payload encriptado, no sobre el paquete IP completo.

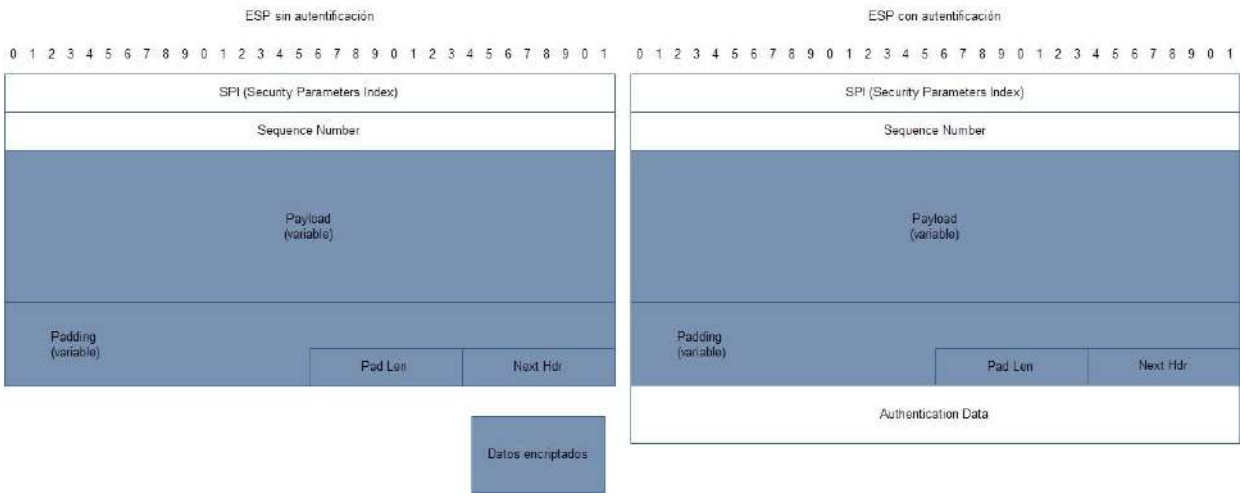


Figura 2.6.4: Encabezado ESP sin y con Autentificación.

2.6.2.1. ESP en Modo de Transporte

Al igual que en AH, el modo transporte encapsula solo al payload del datagrama y está diseñado estrictamente para comunicaciones host-a-host. El encabezado IP original queda en su lugar, excepto por el campo proto cuyo valor es copiado al campo Next Hdr y es modificado para indicar que le seguirá el encabezado ESP, tal como se muestra en la Figura 2.6.5.

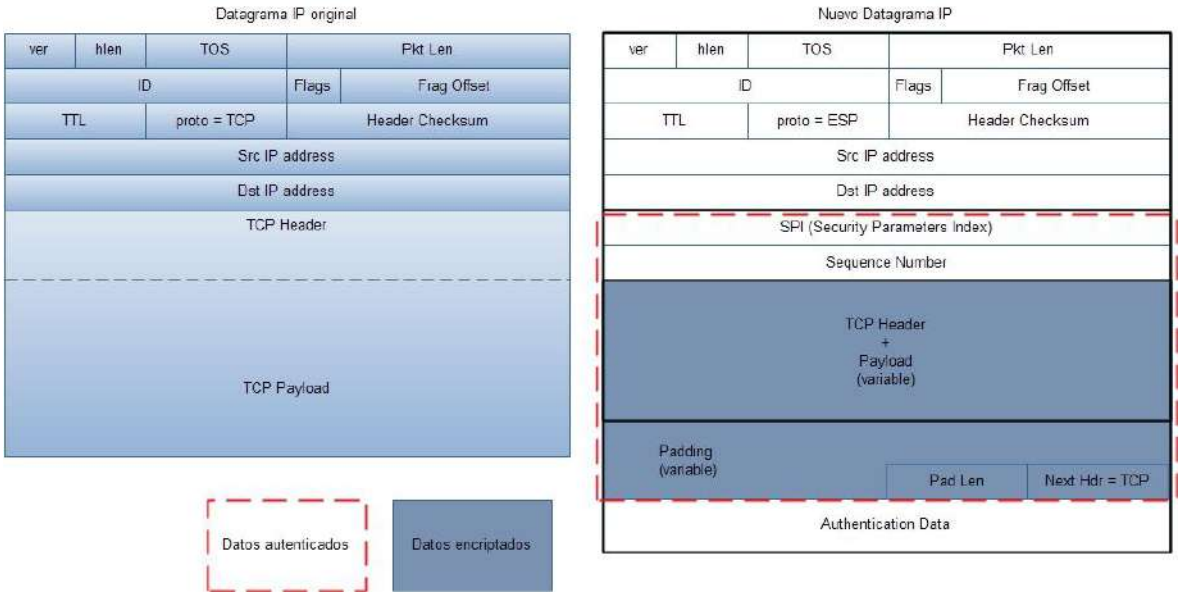


Figura 2.6.5: IPsec en ESP modo transporte.

2.6.2.2. ESP en Modo de Túnel

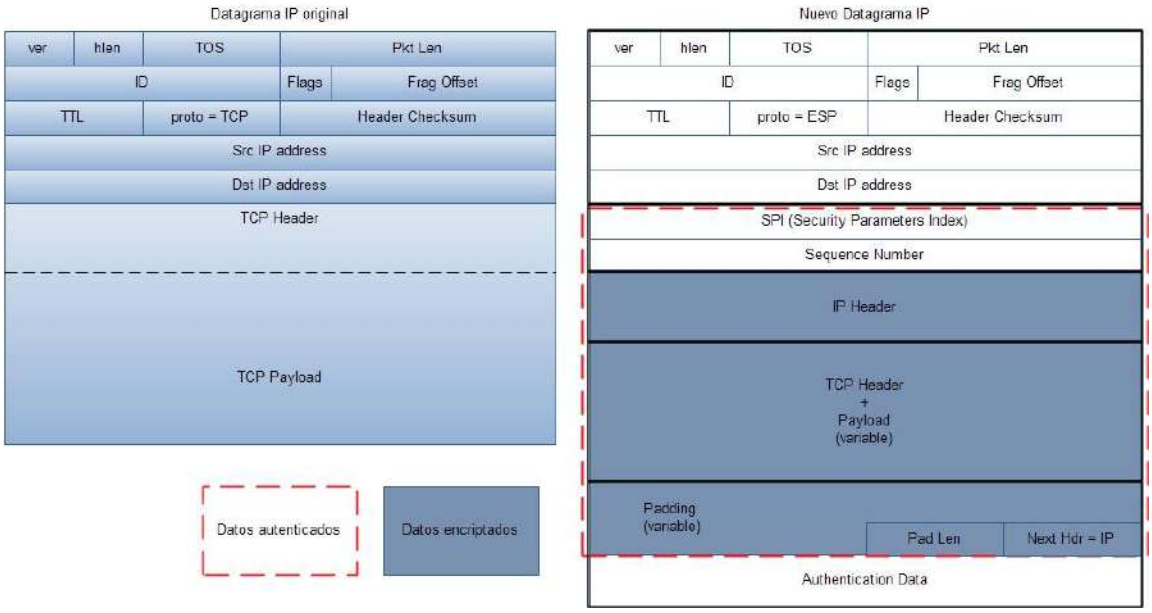


Figura 2.6.6: IPsec en ESP modo túnel.

Como es de suponerse, el modo túnel (Figura 2.6.6), el datagrama IP es encapsulado completamente dentro de otro. La diferencia más notable con AH, es que determinar el modo que se utilizó para transportar un paquete no puede ser determinado por medio de una inspección

poco profunda debido a que el campo Next Hrd es parte del payload encriptado, y no puede ser visto por nadie que no pueda desencriptar al paquete.

2.6.3. Asociaciones de Seguridad (Security Association o SA)

Una Asociación de Seguridad (SA) es una colección de parámetros de seguridad para una conexión específica. Cuando un datagrama arriba, el host o el Gateway deben inspeccionar en una base de datos específica para determinar que SA le corresponde, y así saber qué servicios de seguridad debe aplicarle o cuales le fueron aplicados. El resultado de esto, le permite al emisor construir un nuevo datagrama IP (datagrama IPSec) dándole todos los servicios que fueron asignados a la hora de definir una política, y le permite al extremo final realizar todas las comprobaciones que sean necesarias y reconstruir el datagrama original.

Todas las implementaciones de AH o ESP deben soportar el concepto de SA. A su vez, una SA estará relacionada a una única protección, es decir, AH ó ESP, pero no a ambas. Si un tráfico fue protegido por ambas protecciones, entonces se necesitarán dos o más SAs distintas para ofrecer protección a dicho flujo. Además, las SA son aplicables un solo sentido, de entrada o de salida. Para proteger a una conexión full-duplex deberán crearse, por separado, SAs para el tráfico entrante y SAs para el tráfico saliente.

En ocasiones, puede ocurrir que un flujo en particular necesite la protección de varios protocolos para cumplir con una determinada política de seguridad. Debido a lo mencionado anteriormente, esto no puede ser logrado mediante el uso de una única SA. En estos casos, será necesario emplear varias SAs para alcanzar el objetivo solicitado. La combinación de estas asociaciones de seguridad puede realizarse de dos maneras:

- **Adyacencia de transporte (Transport adjacency):** La adyacencia de transporte refiere a la aplicación de más de un protocolo de seguridad al mismo datagrama IP, sin invocar a túneles. Este enfoque para combinar AH y ESP permite un solo nivel de combinación; ya que más combinaciones no agregan beneficios debido a que el procesamiento es realizado por una única instancia de IPsec en el último destino.

- **Iteración de túneles (iterated tunneling):** Iteración de túneles refiere a la aplicación de varias capas de protocolos de seguridad realizados sobre un túnel IP. Este método permite múltiples niveles de anidamiento, ya que cada túnel puede originarse o terminar en distintos sitios de IPsec a lo largo del camino.

Para la adyacencia de transporte solo existe un escenario posible, ilustrado por la Figura 2.6.7, en cambio, para la iteración de túneles, pueden darse tres casos básicos.

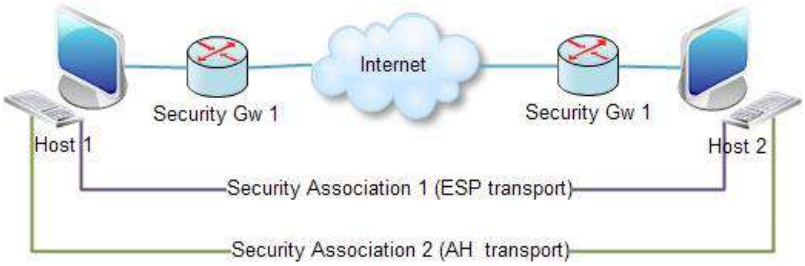


Figura 2.6.7: Adyacencia de Transporte (Caso 1).

En el primer caso para iteración de túneles, ambos extremos son iguales para los SAs. El túnel interno y el externo pueden ser AH o ESP. Por lo general, cada túnel aplica un protocolo distinto, aplicar el mismo protocolo a ambos túneles es poco probable y raramente útil. En el segundo caso que podemos plantear, uno de los extremos coincide para las SAs. En el último caso, ninguno de los extremos coincide para las SAs. Las Figura 2.6.8, Figura 2.6.9, y Figura 2.6.10 muestran estos casos.

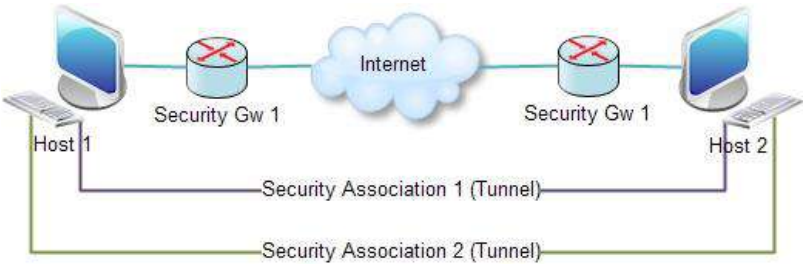


Figura 2.6.8: Iteración de Túneles (Caso 1).

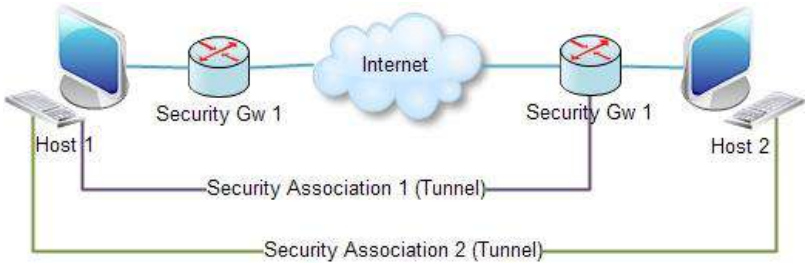


Figura 2.6.9: Iteración de Túneles (Caso 2).

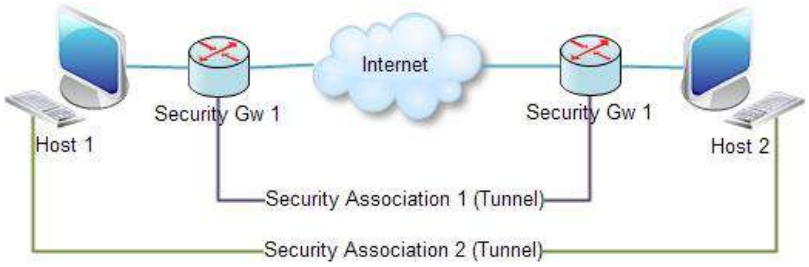


Figura 2.6.10: Iteración de Túneles (Caso 3).

Existen dos bases de datos muy importantes para el procesamiento de tráfico IP en relación a SAs, la base de datos de políticas de seguridad (Security Policy Database o SPD) y la base de datos de asociaciones de seguridad (Security Associations Database o SAD). La primera, especifica las políticas que determinan la disposición de todo el tráfico IP entrante o saliente de un host, Gateway seguro, o implementación de IPsec. La última, contiene los parámetros que son asociados con cada asociación de seguridad.

La SPD debe discriminar entre una gran cantidad de tráfico, decidiendo a qué flujos se les debe brindar protección IPsec y a cuáles no. Esta discriminación se aplica tanto en el emisor como en el receptor. Para cualquier datagrama entrante o saliente, existen tres opciones de procesamiento:

- **Descartar:** Cualquier tráfico que no tenga permitido existir en un host, atravesar un Gateway seguro, o que no se debe entregar a una aplicación en absoluto, debe ser descartado.

- **Saltar IPsec:** El tráfico que no requiere ninguna protección IPsec, debe ser entregado sin ninguna encapsulación o encabezado adicional.

- **Aplicar IPsec:** El tráfico que requiera protección IPsec debe ser tratado por la instancia de IPsec para aplicarle todo el procesamiento necesario. Para esto, la SPD debe especificar los servicios de seguridad que se deben proveer, los protocolos a ser empleados, los algoritmos a utilizar, etc.

La SPD contiene una lista ordenada de entradas de políticas. Cada entrada de política, está indexada para uno o más selectores que definen el conjunto de tráfico IP abarcada por esa dicha entrada. Esto define una granularidad de políticas o SAs. Cada entrada incluye una indicación de si el tráfico coincidente con esta política será pasado por alto, descartado, o sujeto a procesamiento IPsec. Debido a que una política de seguridad puede requerir de más de una SA para ser aplicada a un tráfico, en un orden especifico, la entrada de la política en la SPD debe preservar dicho ordenamiento de requerimientos. De esta forma, la implementación IPsec podrá determinar qué paquete de entrada o de salida debe ser procesado a través de una secuencia de SAs.

Cada entrada de la SAD define los parámetros asociados con una SA. Cada SA tiene una entrada en la SAD. Para el procesamiento saliente, las entradas de la SAD son apuntadas por entradas de la SPD. Si una entrada del SPD no apunta a una SA que sea apropiada para el paquete, la implementación de IPSec crea una SA o conjunto de SAs apropiadas y enlaza la entrada SPD a la entrada SAD. Para el procesamiento entrante, cada entrada en la SAD es indexada por la dirección IP destino, el tipo de protocolo IPsec (AH o ESP), y el índice de parámetros de seguridad (SPI).

2.6.4. Administración de claves (Key Management)

Las facilidades criptográficas de autentificación y encriptación, brindadas por IPsec, requieren del uso de claves secretas únicamente conocidas por los participantes y por nadie más. La forma más sencilla de establecerlas es por medio de una configuración manual. Un participante genera un conjunto de claves y las transmite al resto. Sin embargo, esto es poco escalable e inseguro, a mayor número de dispositivos utilizando las mismas claves, existe una mayor posibilidad de fuga de este secreto, y la redistribución de nuevas claves se hace más compleja.

La forma más segura de manejar claves secretas es a través de IKE (Internet Key Exchange), el cual permite a dos extremos establecer sus asociaciones de seguridad, incluyendo a las claves secretas a ser utilizadas por éstas. IKE utiliza a ISAKMP (Internet Security Association Key Management

Protocol) como framework para dar soporte al establecimiento de SA compatibles entre los extremos.

2.7. Resumen

En este capítulo se vieron las diferentes formas para la implementación de calidad de servicio, ya sea por el marcado de los paquetes para la aplicación de distintas políticas, el control de congestión en una comunicación orientada a la conexión y la introducción de métodos de seguridad para lograr la confiabilidad de los datos transportados.

DiffServ es una arquitectura de red que especifica un mecanismo simple, escalable y finamente granulado para clasificar y administrar tráfico de red y proveer garantía de calidad de servicios en redes IP modernas. Puede ser usado para proveer baja latencia, servicios garantizados a tráfico de red crítico tales como voz o video, mientras provee también garantía de mejor esfuerzo a servicios no críticos como tráfico web o transferencia de archivos. Utiliza los seis bits del campo DSCP del encabezado IP para los propósitos de clasificación reemplazando el campo IP Precedence que originalmente era utilizado para este propósito.

Una ventaja de Diffserv es que el establecimiento de políticas y la clasificación son hechos en los bordes del dominio. Esto significa que en el núcleo de Internet, los routers pueden concentrarse en el trabajo de encaminamiento y despreocuparse acerca de las complejidades de los acuerdos de calidad. Tampoco requiere una configuración avanzada ni de reservación, y no consume tiempo en la negociación extremo-a-extremo para cada flujo, como ocurre en Servicios Integrados. Esto hace a DiffServ relativamente sencillo de implementar.

La pérdida de paquetes es un hecho habitual en Internet debido a las limitaciones de memoria de los buffers de los routers y la gran cantidad de tráfico que viajan a través de ellos. Muchas veces el caudal de tráfico entrante en un router es mayor al caudal de paquetes que este puede despachar. Sumado a esto, los protocolos de transporte orientados a la conexión pueden generar más tráfico con la retransmisión de los paquetes perdidos.

ECN es una de las alternativas que existen para reducir la cantidad de paquetes perdidos y la congestión en los routers. Los routers pueden detectar la presencia de congestión antes de que

ésta realmente ocurra, de manera que puede notificarle a los extremos de la comunicación que reduzcan la cantidad de paquetes a transmitir y reducir así la pérdida y retransmisión de paquetes.

La notificación de congestión es realizada a través de IP en los bits ECN del campo DS y la notificación de reducción se realiza a través banderas establecidas en el campo reservado del encabezado TCP. De esta manera, no es necesario adoptar un nuevo protocolo y el uso de estas banderas no requieren de un gran procesamiento por parte de los nodos intervinientes en la transmisión.

Capítulo 3: Herramientas para la calidad de servicio en GNU/Linux

3.1. Introducción

El término calidad de servicio o QoS (Quality of Service) refiere a la garantía de entrega y un correcto tratamiento de los datos que son transportados a través de un domino. Esta garantía se lleva a cabo a través de un conjunto de protocolos y tecnologías que le aseguran a las aplicaciones que requieran de características especiales en sus flujos de datos (como baja latencia o una mayor asignación de ancho de banda) disponer de los recursos necesarios para hacer cumplir estas necesidades de calidad. Una de las principales metas de la calidad de servicio es la priorización de los flujos, donde se garantiza el envío de aquellos flujos que poseen mayores requisitos frente a otros de menores requisitos.

Existe diversas formas de implementar calidad de servicio en un entorno GNU/Linux. Una de ellas es a través de la manipulación de los paquetes que llegan al Kernel de Linux mediante Netfilter. Netfilter es un Framework disponible en el Kernel que permite interceptar y manipular paquetes de red en los distintos estados del procesamiento de los mismos. El componente más popular construido sobre Netfilter es *iptables*, una herramienta esencialmente de firewall que también permite realizar una traducción de direcciones de red (NAT), así como también mantener un registro de los paquetes que arriban al Kernel.

Otra herramienta útil para la implementación de QoS es *tc* (traffic control) que permite establecer y administrar políticas de calidad, clasifica el tráfico de entrada y salida según sus características y lo asigna a una política previamente establecida. Esta herramienta está incluida dentro iproute2, un conjunto de herramientas para control de redes TCP/IP y control de tráfico en Linux.

Con la combinación de estas dos herramientas recién mencionadas puede establecerse un mecanismo para el control de tráfico. Por ejemplo, marcando según requerimientos específicos (valor de DSCP, prioridades, etc.) los paquetes de un flujo de datos en particular que se caracteriza a través de la dirección IP destino, puerto destino, protocolo de transporte, etc.; y aplicando políticas y disciplinas de colas específicas para que los paquetes con alguna marca en especial (por

ejemplo el valor del campo DSCP) sean tratados con mayor prioridad y sean enviados antes que el tráfico común o de menor prioridad.

3.2. Procesamiento de paquetes TCP/IP en el Kernel de Linux

Antes de comenzar a explicar el funcionamiento de las herramientas utilizadas para establecer QoS en un entorno GNU/Linux es necesario entender cómo es que el Kernel Linux procesa los paquetes TCP/IP (Balliache, 2003).

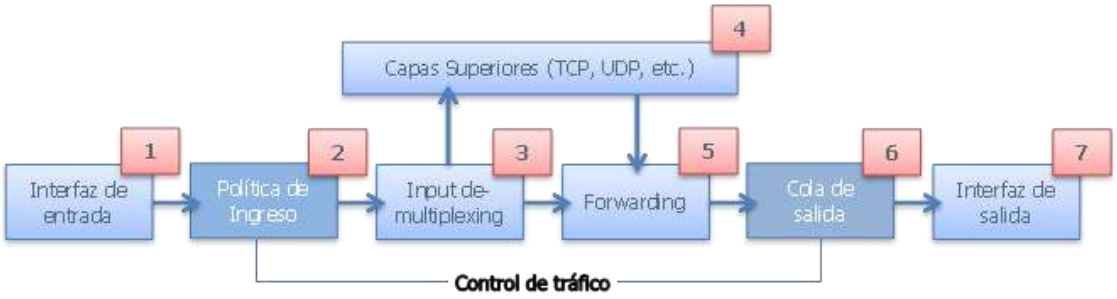


Figura 3.2.1: Procesamiento de paquetes en el Kernel Linux.

Todos los paquetes que provienen de la interfaz de red (Figura 3.2.1 - 1) o que son generados por la propia máquina son procesados por el Kernel de Linux. El primer paso consiste en examinar si los paquetes que llegan al de-multiplexor de entrada (Figura 3.2.1 - 3) están destinados a algún proceso local. De ser así, estos son enviados a una capa superior (Figura 3.2.1 - 4) para ser procesados. De lo contrario, los paquetes son enviados al bloque de reenvío o forwarding (Figura 3.2.1 - 5). Este bloque, que también puede recibir paquetes generados localmente desde una capa superior (Figura 3.2.1 - 4), determinará el próximo salto del paquete a partir de la tabla de ruteo. Después de esto, los paquetes son encolados en la cola de salida (Figura 3.2.1 - 6) para ser transmitidos por la interfaz de salida (Figura 3.2.1 - 7). Es en este punto donde entra en juego el control de tráfico, el cual puede ser usado para construir una compleja combinación de disciplinas de colas, clases y filtros que controlan como los paquetes deben ser enviados a través de la interfaz de salida.

También es posible aplicar un control de tráfico en la interfaz de entrada, antes del de-multiplexor de entrada (Figura 3.2.1 - 2). Esto sirve para descartar paquetes indeseados aliviando

procesamiento innecesario y liberando memoria, marcar paquetes, traducir direcciones, etc.

3.3. Framework Netfilter

Como se menciona en el Proyecto Netfilter (Rusty Russell, 2002), su arquitectura es simplemente una serie de estados (*hooks*) en varios puntos en la pila de protocolos (por ejemplo IPv4, IPv6 y DECnet) en que se encuentran los paquetes. Los estados de Netfilter que pueden ser vistos en la Figura 3.3.1 son:

1. **PREROUTING (NF_IP_PRE_ROUTING):** Todos los paquetes pasan por este estado, sin excepciones. Este es alcanzado antes que la decisión de ruteo sea tomada y después de que se ha verificado el encabezado IP. En este estado se implementa Network Address Port Traslation (NAPT) y Destination Network Address Traslation (DNAT).

2. **LOCAL INPUT (NF_IP_LOCAL_IN):** Todos los paquetes que tienen como destino la máquina local llegan a este estado. Es el último estado del tráfico entrante.

3. **FORWARD (NF_IP_FORWORD):** Los paquetes que no son destinados a la máquina local alcanzan este estado.

4. **LOCAL OUTPUT (NF_IP_LOCAL_OUT):** Es el primer estado en el camino de salida de los paquetes. Los paquetes que son enviados desde la máquina local siempre pasan por este estado.

5. **POSTROUTING (NF_IP_POST_ROUTING):** Este estado es alcanzado después de la decisión de ruteo. En este se lleva a cabo el Source Network Address Traslation (SNAT), y todos los paquetes que pasan por la máquina local pasan por este estado.

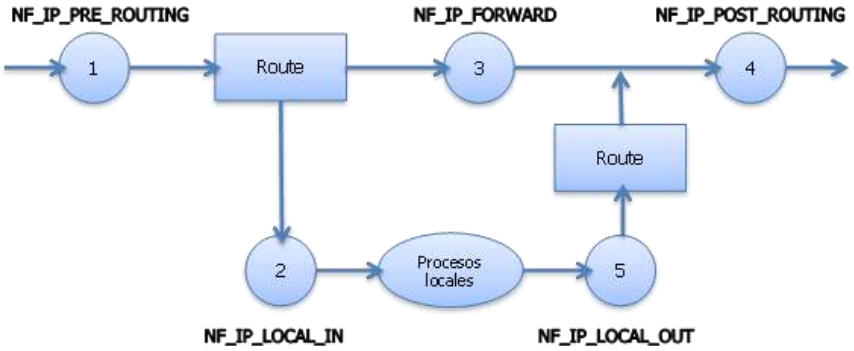


Figura 3.3.1: Estados de Netfilter.

A partir de las definiciones de los estados de Netfilter se puede diferenciar tres tipos distintos de flujos de tráfico dependiendo del destino de los datos.

- Tráfico que pasa a través de la máquina, es decir, que no está destinado a la máquina local. Este tráfico sigue el siguiente camino: PREROUNTING -> FORWARD -> POSTROUTING.

- Tráfico entrante, es decir, destinado a la máquina local. Sigue el camino: PREROUTING -> LOCAL INPUT.

- Tráfico saliente, es decir, el tráfico originado por la máquina local. El camino que sigue es: LOCAL OUTPUT -> POSTROUTING.

Se pueden registrar módulos del Kernel para escuchar en cualquiera de estos estados. Un módulo que registra una función debe especificar la prioridad de esta función dentro del estado, por lo tanto cuando este estado de Netfilter es llamado, cada módulo registrado hasta ese punto es invocado en función de la prioridad y es libre de manipular el paquete. El módulo puede entonces decirle a Netfilter que realice alguna de estas cinco acciones:

1. **Aceptar (NF_ACCEPT):** Dejar que el paquete siga viajando a través de la pila.

2. **Descartar (NF_DROP):** Eliminar silenciosamente el paquete.

3. **Retener (NF_STOLEN):** Se retiene temporalmente el paquete hasta que algo sucede, de

forma que no continúa su trayecto a través de la pila. Es usado generalmente para recolectar paquetes IP fragmentados.

4. **Encolar (NF_QUEUE):** Encola el paquete, envía el paquete al espacio de usuario para que pueda ser modificado por alguna aplicación.

5. **Repetir (NF_REPEAT):** Fuerza al paquete volver nuevamente al mismo estado.

En resumen, Netfilter provee métodos para registrar llamadas a funciones que realicen algún tipo de manejo de los paquetes en cualquiera de los estados definidos por el framework. El valor devuelto por las funciones, indicando alguna de las cinco acciones mencionadas anteriormente, es tomado por el framework el cual aplicará la política deseada basada en el resultado obtenido.

3.3.1. Iptables

Iptables es uno de los componentes del framework de Netfilter extendido enormemente a partir de las series de Kernels 2.4.x y 2.6.x, sucesor de los sistemas *ipchains* de la serie 2.2.x y de *ipfwadm* de la serie 2.0.x. Es una aplicación definida en el espacio de usuario usado para configurar reglas de filtrado de paquetes. Está estructurada en forma de tabla genérica de definiciones de reglas, donde cada regla dentro una tabla IP consiste de un número de clasificadores (iptables matches) y una acción relacionada (iptables target).

Los módulos del Kernel pueden registrar una nueva tabla y pedir por un paquete que atraviese por una tabla dada. Este método de selección de paquetes se utiliza para el filtrado de paquetes (tabla “filter”), la traducción de direcciones de red (tabla “nat”) y para manipulación de paquetes pre-ruteados en general (taba “mangle”).

3.3.1.1. Tablas Iptables

Las tablas definidas por el Framework Netfilter para iptables son las siguientes:

Filter*:* Esta tabla debería ser utilizada específicamente para aceptar o rechazar paquetes. Los objetivos (o targets) que pueden utilizarse sin ningún tipo de problema para las reglas en esta tabla son: DROP, LOG, ACCEPT o REJECT.

Esta tabla es atravesada por los paquetes que se encuentran en los estados:

- LOCAL INPUT.
- FORWARD.
- LOCAL OUTPUT.

NAT*:* Esta tabla es utilizada con el fin de administrar Network Address Translation (NAT). Los paquetes "nateados" tienen sus direcciones IP alteradas, de acuerdo a una regla. Los paquetes pertenecientes a un flujo pasan por esta tabla solo una vez. Asumiendo que el primer paquete de un flujo es aceptado, el resto de los paquetes en el mismo flujo son automáticamente "nateados" o enmascarados, y serán dirigidos por las mismas acciones que al primero. De esta manera, los paquetes no pasan nuevamente por esta tabla, sin embargo son tratados como al primer paquete del flujo. Es por esta razón que el filtro no debe realizarse en esta tabla. En esta tabla pueden utilizarse los siguientes objetivos: DNAT, SNAT y MASQUERADE.

Esta tabla es atravesada por los paquetes que se encuentran en los estados:

- PREROUTING.
- LOCAL OUTPUT.
- POSTROUTING.

Mangle*:* Esta tabla es utilizada principalmente para el manejo de paquetes. Entre otras cosas, se pueden cambiar los contenidos de los diferentes paquetes y sus encabezados. Ejemplos de esto puede ser cambiar los campos TTL, TOS o MARK. Hay que notar que MARK no es realmente un cambio en el paquete, pero si un valor de marca para el paquete que se establece en el espacio de Kernel. Otras reglas o programas podrían usar esta marca más adelante en un firewall para filtrar o realizar un ruteo avanzado, tc es un ejemplo.

Esta tabla es atravesada por los paquetes que se encuentran en los estados:

- PREROUTING.
- LOCAL INPUT.
- FORWARD.

- LOCAL OUTPUT.
- POSTROUTING.

3.3.1.2. Sintaxis de Iptables

Cada regla es una linea que el Kernel observa para encontrar que hacer con un paquete. Si todos los criterios (matches) se cumplen, se lleva a cabo la instrucción objetivo (target ó jump). Generalmente las reglas de *iptables* se escriben con una sintaxis similar a la siguiente:

```
iptables[-t tabla] comando cadena [match] [target/jump]
```

No es necesario que la instrucción target sea la última función en la línea, sin embargo, generalmente se adhiere a esta sintaxis para tener una mejor lectura.

El parámetro *tabla* puede tomar como valor cualquiera de las tablas que se definieron en la sección 3.3.1.1, aunque no es necesario especificar este parámetro. De no indicarse una tabla, por defecto iptables coloca la regla en la tabla filter.

El parámetro comando específica a iptables que hacer con el resto de la regla, generalmente lo que se desea hacer es agregar o eliminar una regla en una tabla. Los comandos disponibles son:

- **-A | --append:** Agrega una regla al final de la cadena especificada.

- **-D | --delete:** Elimina una regla de la cadena especificada. Se puede realizar de dos maneras, indicando en número de regla dentro de le la cadena (comenzando de 1 desde la primer regla) o ingresando la regla completa que debe coincidir exactamente con la regla de la cadena que se desea eliminar.

- **-R | --replace:** Reemplaza una entrada en la cadena especificando el número de regla. Funciona de forma similar al comando delete pero en lugar de eliminar la regla la reemplaza por otra nueva. Su principal uso es para experimentación.

- **-I | --insert:** Inserta una nueva regla en la cadena en el lugar especificado por el número de regla.

- **-L | --list:** Este comando lista todas las reglas de una tabla y cadena especificada. Si no se especifica una cadena el comando listará todas las posibles para la tabla.

- **-F | --flush:** Elimina todas las reglas de una cadena y una tabla especificada. De no especificarse una cadena en particular, el comando elimina todas las reglas de todas las cadenas.

- **-Z | --zero:** Este comando indica al programa que reinicie todos los contadores de una cadena especificada o de todas las cadenas de una tabla.

- **-N | --new-chain:** Este comando le indica al Kernel crear una nueva cadena con el nombre especificado en la tabla especificada.

- **-X | --delete-chain:** Este comando elimina una cadena especificada de la tabla. No debe haber ninguna regla que refiera a esa cadena para poder ser eliminada. Si no se especifica ninguna opción elimina todas las cadenas creadas por el usuario para la tabla especificada.

- **-P | --policy:** Le indica al Kernel establecer un target o política por defecto a una cadena.

- **-E | --rename-chain:** Renombra la cadena especificada con el nuevo nombre recibido por parámetro.

El parámetro cadena (o chain) se correlaciona con los estados o hooks determinados por el framework de Netfilter. De esta manera, el parámetro cadena indica el estado válido del paquete para la tabla referenciada por el comando *table*. Por ejemplo, para la tabla NAT el valor que puede tomar este parámetro es PREROUTING, LOCAL OUTPUT ó POSTROUTING.

Los parámetros *match* y *target* o *jump* serán tratados de forma separada a la sintaxis del comando y se desarrollaran en las siguientes secciones.

3.3.1.3. Matches

Los matches son la parte de la regla enviados al Kernel que contienen detalles específicos de las características del paquete, es decir, lo que permite diferenciar un paquete de otro e identificarlos. A través de este parámetro puede especificar la dirección IP de la cual proviene el paquete, desde que interfaz de red arriba, la dirección IP destino, el puerto, protocolo, etc. Existen varios tipos de *matches* que pueden ser utilizados, de los cuales algunos son mencionados a continuación. Por cuestiones de organización primero se van a mencionar los *matches* que son utilizados para propósito general, luego los que son específicos para determinados protocolos (TCP, UDP e ICMP respectivamente) y finalmente los de propósitos especiales.

Matches de propósito general: Estos matches están siempre disponibles sin importar la clase de protocolo sobre el cuál se está trabajando o qué tipo de extensiones fueron cargadas. Son utilizados sin la necesidad de un parámetro en particular.

- **-p | --protocol:** Se utiliza para evaluar por algún protocolo en particular. Ejemplos de estos protocolos son TCP, UDP e ICMP.

- **-s | --src | --source:** Es el match fuente, el cual es utilizado para evaluar paquetes basándose en su dirección IP fuente. Se puede indicar tanto la dirección IP como su máscara de red.

- **-d | --dst | --detination:** Es el match destino, el cual es utilizado para evaluar paquetes basándose en su dirección IP destino. Se puede indicar tanto la dirección IP como su máscara de red.

- **-i | --in-inteface:** Es utilizado para evaluar la interfaz por la cual ingresa el paquete. Solo puede ser utilizado en las cadenas LOCAL INPUT, FORWARD o PREROUTING, si es usado en cualquier otra cadena devolverá un error.

- **-o | --out-inteface:** Es utilizado para evaluar la interfaz por la cual egresa el paquete. Solo puede ser utilizado en las cadenas LOCAL OUTPUT, FORWARD o POSTROUTING, si es

usado en cualquier otra cadena devolverá un error.

- **-f | --fragment:** Es utilizado para evaluar los fragmentos de un paquete fragmentado.

TCP Matches: Estos matches son específicos del protocolo TCP y solo están disponibles cuando se trabajan con flujos y paquetes TCP. Para poder usarlos es necesario especificar ***–p tcp*** en la regla antes de usarlos.

- **--sport | --source-port:** Es utilizado para identificar paquetes en base al número de puerto origen. Si no se especifica se consideran implícitamente todos los puertos. También puede utilizarse el nombre de un servicio que debe estar definido en /etc/services. Este parámetro puede ser también utilizado para establecer rangos de puertos.

- **--dport | --destination-port:** Es utilizado para identificar paquetes en base al número de puerto destino. Si no se especifica se consideran implícitamente todos los puertos. También puede utilizarse el nombre de un servicio que debe estar definido en /etc/services. Este parámetro puede ser también utilizado para establecer rangos de puertos.

- **--tcp-flags:** Es utilizado para identificar las banderas del paquete TCP. A este parámetro se le especifican dos listas de banderas. La primera lista las banderas a comparar (mascara) y la segunda lista aquellas banderas que deberían estar establecidas en 1. Las banderas que se puede usar son: SYN, ACK, FIN, RST, URG y PSH. También se puede utilizar ALL para indicar todas las banderas, y NONE para indicar ninguna.

- **--syn:** Es utilizado para identificar paquetes que tienen el bit SYN establecido y los bits ACK y RST en cero. En otras palabras es el equivalente a *–tcp-flags* SYN, ACK, RST SYN. Estos paquetes son utilizados principalmente para el inicio de conexión TCP.

- **--tcp-option:** Es utilizado para identificar paquetes basándose en las opciones de TCP en sus valores decimales, que son una parte específica del encabezado.

UDP Matches: Estos matches son específicos del protocolo UDP y solo están disponibles cuando se trabajan con paquetes UDP. Para poder usarlos es necesario especificar ***–p udp*** en la regla antes de usarlos. Dado a que UDP no es orientado a la conexión, no existen diferentes banderas para indicar inicio y cierre de conexión como en TCP. Tampoco existen confirmaciones de recibo de paquetes (ACKs). Por estos motivos es que existen menos matches.

- **--sport | --source-port:** Es utilizado para identificar paquetes en base al número de puerto origen. Si no se especifica se consideran implícitamente todos los puertos. También puede utilizarse el nombre de un servicio que debe estar definido en /etc/services. Este parámetro puede ser también utilizado para establecer rangos de puertos.

- **--dport | --destination-port:** Es utilizado para identificar paquetes en base al número de puerto destino. Si no se especifica se consideran implícitamente todos los puertos. También puede utilizarse el nombre de un servicio que debe estar definido en /etc/services. Este parámetro puede ser también utilizado para establecer rangos de puertos.

ICMP Matches: Estos matches son específicos del protocolo ICMP y solo están disponibles cuando se trabajan con paquetes ICMP. Para poder usarlos es necesario especificar ***–p icmp*** en la regla antes de usarlos. Este protocolo es principalmente usado para reportar errores y controlar conexiones entre otros usos. La principal característica de este protocolo es el encabezado type que específica para qué es el paquete.

- **--icmp-type:** Es utilizado para especificar el tipo de ICMP. Puede ser especificado por el valor numérico tanto como por su nombre según el estándar RFC 792 (Postel, 1981).

Matches de propósitos especiales: Los matches de propósitos especiales deben ser específicamente cargados con la opción ***--m*** o ***--match*** para poder ser utilizados. Mientras que algunos de estos matches pueden ser específicos de algún protocolo, otros pueden ser usados con cualquier protocolo. Ejemplos de estos es el match state, utilizado para el seguimiento de conexiones.

Match limit: La extensión limit es cargada explícitamente con la opción ***–m limit***. Este match es

utilizado para limitar la cantidad de veces que una regla puede ocurrir en un intervalo de tiempo. Puede utilizarse por ejemplo para evitar ataques de denegación de servicio, o para evitar ataques de fuerza bruta sobre una conexión SSH. Las opciones que pueden utilizarse con este match son:

- **--limit:** Establece la tasa promedio máxima de ocurrencia. Se especifica con un número y opcionalmente con una unidad de tiempo (/second, /minute, /hour, /day). Esto indica la cantidad de veces que puede ocurrir el match por unidad de tiempo.

- **--limit-brust:** Sirve para establecer la ráfaga límite del match. Le indica a iptables el número máximo número de paquetes que pueden coincidir con la regla dentro de una unidad de tiempo dada.

Match MAC: Puede utilizarse para identificar paquetes según su dirección MAC (Ethernet Media Access Control) de origen. Para indicar el uso de este match en una regla se utiliza: ***–m mac --mac-source***. La dirección debe estar escrita de la forma XX:XX:XX:XX:XX:XX, donde XX es un valor hexadecimal. Solo puede ser utilizado en las cadenas PREROUTING, FORWARD y LOCAL INPUT.

Match Mark: Esta extensión es utilizada para identificar paquetes basados en las marcas que estos tienen establecidas. Una marca es un campo especial mantenida solo dentro del Kernel, es decir, asociar los paquetes mientras estos viajan a través de la máquina. Esta marca puede ser utilizada por diferentes rutinas del Kernel para tareas como filtrado y control de tráfico. Para indicar el uso de este match en una regla se utiliza: ***-m mark --mark***.

Match Multiport: Puede utilizarse para identificar paquetes según el puerto destino o un rango de puertos destinos. No se puede utilizar el match estándar port y multiport simultáneamente en una regla. Si se especifican ambas en una misma regla, iptables aceptará la primera que aparezca e ignorará la otra.

- **--source-port:** Este match indica múltiples puertos de origen. Pueden especificarse hasta 15 puertos separados por comas y solamente puede utilizarse para los protocolos TCP y UDP.

- **--destination-port:** Este match indica múltiples puertos destino.

- **--port:** Es utilizado para indicar paquetes en base al puerto fuente como destino.

Match Owner: Puede utilizarse para identificar paquetes basados en la identidad del proceso que lo creo. El dueño puede ser especificado tanto por el ID de proceso como el usuario que ejecuto el comando en cuestión, el ID del grupo, el proceso, o la sesión. Solo puede ser utilizada en la cadena OUTPUT debido a que es imposible encontrar información acerca de la identidad de la instancia que envió el paquete desde otro nodo.

- **--uid-owner:** Sirve para identificar los paquetes que fueron creados por el User ID (UID) dado.

- **--gid-owner:** Sirve para identificar los paquetes que fueron creados por el Group ID (GID) dado.

- **--pid-owner:** Sirve para identificar los paquetes que fueron creados por el Process ID (PID) dado.

- **--sid-owner:** Sirve para identificar los paquetes que fueron creados por el Session ID (SID) dado.

Match State: Esta extensión es utilizada en conjunto con el código de seguimiento de conexión (connection tracking - Conntrack) del Kernel. El match state accede a los estados de conexión de los paquetes desde la máquina de estados. Esto permite conocer en cuál de los estados de la conexión se encuentra el paquete y funciona para muchos protocolos, incluyendo protocolos sin estados como ICMP o UDP. En todos estos casos, existirá un *timeout* por defecto para la conexión y luego será descartado de la base de datos de los seguimientos de conexiones. Este match necesita ser cargado explícitamente mediante la sentencia ***–m state*** en la regla. El concepto de máquina de estado y su uso se verá con más detalle en la sección 3.3.2 Conntrack.

- **--state:** Esta opción establece el estado en que se debe encontrar el paquete para ser identificado por la regla. Puede tomar cuatro valores posibles: INVALID, NEW,

ESTABLISHED y RELATED.

Match TOS: Es utilizado para identificar paquetes basados en su octeto TOS (Type Of Service) del encabezado IP. Este match debe ser cargado mediante ***-m tos***. El significado y uso que hacen los routers del valor contenido en este campo ya fue explicado anteriormente en la sección 2.3.2.

- **--tos:** Utilizado para lo descripto anteriormente. Puede recibir valores hexadecimales o decimales.

Match TTL: Es utilizado para identificar paquetes según su tiempo de vida (TTL), sin realizar cambios en el paquete. Este match necesita ser cargado explícitamente mediante la sentencia ***–m ttl –ttl*** en la regla.

3.3.1.4. Targets o jumps

Los targets o jumps le indican a la regla que hacer con los paquetes que coinciden exactamente con un match en el respectivo parámetro. Para indicar el uso de targets/jumps se debe utilizar –j target/jump en la regla. Los targets indican la acción directa sobre el paquete, por ejemplo aceptarlo (ACCEPT) o rechazarlo (DROP). En cambio, el término jumps se utiliza para indicar que un paquete identificado por un match debe pasar por otra regla especificada o verificar que el paquete cumple con alguna regla de otra tabla o cadena por defecto de iptables o una tabla o cadena creada por el usuario. Por ejemplo, suponiendo que se crease una tabla especial para el tratamiento de paquetes TCP llamada *tabla_tcp*, el uso de jumps para indicar que los paquetes TCP deben pasar por esta tabla quedaría expresado de la siguiente forma:

```
iptables-A INPUT -p tcp -j tabla_tcp
```

Los paquetes son comparados con cada una de las reglas hasta tener una coincidencia exacta con un *match*, pero además depende de la acción a realizar con estos paquetes para que sigan siendo o no comparados con el resto de las reglas. Es así, que dentro de los *targets* podría hacerse una clasificación entre los *targets* que detienen la comparación con otras reglas (por ejemplo DROP, REJECT o ACCEPT) y aquellos que permiten que sigan siendo comparados con otras reglas dentro de una *cadena* (por ejemplo LOG, ULOG o TOS). A continuación se listan los posibles *targets* que

pueden ser utilizados con *iptables*:

- **ACCEPT:** Los paquetes especificaciones a través del match son aceptados. De esta forma, no siguen siendo comparados con otras reglas de la cadena, otras cadenas o tablas.

- **DNAT:** Este target es utilizado para llevar a cabo Destination Network Address Translation, lo que significa que la dirección IP destino del paquete es modificada por otra. Esta opción está solo disponible para las cadenas PREROUTING y LOCAL OUTPUT en la tabla NAT. La traducción de direcciones se realiza utilizando *--to-destination* IP_desitno luego del target DNAT. También puede hacerse la traducción de puertos destino con este target indicando el puerto luego de la dirección IP de la forma IP_destino:puerto_destino.

- **SNAT:** Este target es utilizado para llevar a cabo Source Network Address Translation, lo que significa que la dirección IP origen del paquete es modificada por otra. Esta opción está solo disponible para la cadena POSTROUTING en la tabla NAT. SNAT, al igual que DNAT, permite la traducción de puertos. La traducción de direcciones se realiza utilizando *--to-source* IP_origen luego del target SNAT.

- **DROP:** Los paquetes especificaciones a través del match son descartados. De esta forma, no siguen siendo comparados con otras reglas de la cadena, otras cadenas o tablas.

- **REJECT:** Lleva a cabo la misma acción que DROP, a diferencia que REJECT indica el motivo por el cuál un paquete fue rechazado a través de un mensaje ICMP. Este mensaje se establece en la regla colocando *--reject-with* luego del target REJECT. Los mensajes válidos son: *icmp-net-unreachable*, *icmp-host-unreachable*, *icmp-port-unreachable*, *icmp-proto-unreachable*, *icmp-net-prohibited* y *icmp-host-prohibited*.

- **MARK:** Es utilizado para establecer marcas Netfilter de valores asociados con paquetes específicos. Este target es válido sólo para la tabla MANGLE. Los valores de este target son utilizados en conjunto con capacidades de ruteo avanzado en Linux (como tc) para enviar diferentes paquetes a través de rutas distintas e indicar que deben utilizarse diferentes

disciplinas de colas (QDISC), etc. para estos paquetes. La marca sobre el paquete se establece colocando *--set-mark* y la marca específica para los paquetes que coincidan con los matches. Esta marca en el paquete es utilizada sólo para realizar una operación dentro del Kernel de la máquina que realiza la marca, con lo cual no se extiende a lo largo del camino del paquete hacia otras máquinas.

- **MASQUERDE:** Este target es utilizado básicamente para el mismo propósito que SNAT, pero no requiere la opción *--to-source*. Esto se debe a que el target fue diseñado para máquinas que obtienen sus direcciones IP de forma dinámica, un ejemplo sería mediante DHCP. De esta forma, en vez de estar cambiando la dirección IP en la regla SNAT por cada vez que la dirección IP cambia, iptables cambia automáticamente la dirección en las reglas que requiere realizar NAT fijándose en la dirección de la interfaz de salida. Cuando esto ocurre, todas las conexiones relacionadas con una dirección anterior son olvidadas. Al igual que SNAT, este target funciona únicamente en la cadena POSTROUTING. Posee la opción *--to-ports* para realizar una traducción de puertos mediante este target.

- **QUEUE:** Utilizado para encolar paquetes en el espacio de usuarios o para una aplicación. Funciona en forma conjunta con otras aplicaciones o programas, ajenos a iptables, como programas avanzados de filtro o de proxy. Si no existen programas que estén atendiendo esta cola de paquetes, el resultado obtenido es el mismo que con el target DROP.

- **REDIRECT:** Re-direcciona los paquetes que son transportados a través de la máquina hacia su propia dirección, es decir, cambia la dirección destino de los paquetes por la dirección de su interfaz de salida. Puede además re-direccionar puertos mediante la opción *--to-ports*. Es válido únicamente para la tabla NAT y las cadenas PREROUTING y LOCAL OUTPUT.

- **RETURN:** Causa que los paquetes que están siendo verificados en una cadena dejen de hacerlo para esta. Si la cadena donde se está controlando el paquete es una sub cadena de otra, el paquete vuelve a la cadena superior para continuar con su verificación con alguna

regla. Si la cadena es una de las principales, como por ejemplo LOCAL INPUT, al paquete se le aplica la política por defecto para esta cadena.

- **TOS:** Este target es utilizado para establecer el valor de TOS en el encabezado IP. A diferencia de MARK, el valor de TOS en el paquete se extiende a lo largo del camino que este recorra. Es válido solo para la tabla MANGLE. Acompañado del target TOS se utiliza la opción *--set-tos* que aplica un valor numérico de 8 bits al campo TOS del paquete, el valor puede ser escrito en decimal (0-255) o en hexadecimal (0x00-0xFF).

- **ECN:** Permite trabajar selectivamente alrededor de los conocidos agujeros negros ECN. Solo se puede utilizar en la tabla MANGLE seguido con la opción *--ecn-tcp-remove* que elimina todos los bits ECN del encabezado TCP. Obviamente solo puede aplicarse a TCP *(-p tcp*).

- **DSCP:** Este target puede utilizarse únicamente en la tabla MANGLE, permite modificar los bits DCSP del campo ToS del encabezado IPv4. Las opciones para este target son *--set-dscp* indicando un valor decimal o hexadecimal, y la opción *--set-dscp-class* para establecer en el campo DSCP una clase de DiffServ.

- **LOG:** Este target activa un registro (log) de información detallada de los paquetes en el Kernel del Linux. De esta forma, se podría realizar un análisis de las reglas para verificar que se estén cumpliendo o un análisis de fallos de la aplicación de las mismas. Puede utilizarse en cualquier tabla y cualquier cadena. Las opciones que pueden utilizarse para este target son: *--log-level* para indicar el nivel de información del registro, *--log-prefix* para agregar un profijo al registro de mensajes con una longitud máxima de 29 caracteres, *--log-tcp-sequence* almacena el número de secuencia de los paquetes TCP junto con la información del registro, *--log-tcp-options* registra las diferentes opciones del encabezado TCP junto con la información del registro*, --log-ip-options* de la misma forma que en TCP registra las opciones del encabezado IP.

- **ULOG:** Provee un registro de eventos en el espacio de usuarios, la información es enviada junto con el paquete hacia un socket de Netlink. De esta forma, uno o más programas en el espacio de usuario reciben el paquete y la información de registro permitiendo un método de seguimiento de los paquetes más completo y sofisticado. Permite, por ejemplo, guardar la información de registros en una base de datos como MySQL o cualquier otra.

3.3.2. Conntrack

Una de las características más importantes construidas sobre el framework de Netfilter es el seguimiento de conexiones (connection tracking), también conocido como máquina de estados (state machine). Este seguimiento de conexiones es implementado en Netfilter mediante el módulo Conntrack, que le permite al Kernel mantener un seguimiento de todas las conexiones de la red y por lo tanto se puede relacionar a todos los paquetes de una conexión. La utilidad principal que se le da a Conntrack es la construcción de firewalls con estados (stateful firewall) ya que permite un nivel de protección mayor a la que brindan los firewalls sin estados (non-stateful firewalls).

Con *iptables*, los paquetes pueden ser relacionados con el seguimiento de conexiones en cuatro estados diferentes conocidos como NEW, ESTABLISHED, RELATED e INVALID que serán explicados más adelante. Mediante el match *–state* puede controlarse fácilmente quién o qué tiene permitido iniciar nuevas conexiones.

Conntrack debe ser cargado como un módulo o como una parte interna misma del Kernel. A su vez, Conntrack puede trabajar con otros módulos que manejan de forma específica protocolos como TCP, UDP o ICMP entre otros. Estos módulos permiten capturar información única de cada paquete permitiendo mantener un seguimiento de cada flujo de datos. Esta información es utilizada por Conntrack para saber en qué estado se encuentra una conexión. Es importante notar a este punto, que el seguimiento de conexiones es completamente independiente del estado de cualquier capa superior tales como los estados de TCP o SCTP. Por esta razón, es posible llevar a cabo el seguimiento de conexión de paquetes pertenecientes a protocolos sin estados, tales como UDP o ICMP, mediante el registro de información de pseudo estado como lo podría ser el tiempo

de expiración (*timeout*) para indicar inactividad y que la conexión sea considerada como finalizada.

En los Kernels anteriores a Netfilter se tenía la posibilidad de activar o desactivar la desfragmentación de paquetes, pero con la incorporación del seguimiento de conexiones esta característica fue eliminada. Esto se debe a que el seguimiento de conexiones no puede trabajar apropiadamente sin la desfragmentación de paquetes y por lo tanto ésta se lleva a cabo automáticamente.

Todos los seguimientos de conexiones son manejados en la cadena PREROUTING a excepción de los paquetes generados localmente, los cuales son manejados en la cadena OUTPUT. Si se genera localmente el paquete inicial de un flujo, el estado de dicha conexión será establecida en NEW dentro de la cadena OUTPUT, y cuando se reciba un paquete de respuesta, el estado cambiará a ESTABLISHED en la cadena PREROUTING. Si el paquete inicial no es generado localmente, el estado NEW será establecido en la cadena PREROUTING. De esta manera, todos los cambios y cálculos son realizados en las cadenas PREROUTING y OUTPUT de la tabla NAT.

3.3.2.1. Las entradas de Conntrack

El seguimiento de conexiones realizado por Conntrack es almacenado en el archivo "*/proc/net/ip_conntrack*" y contiene toda la información necesaria para que el módulo pueda conocer en qué estado se encuentra la conexión. Un ejemplo de cómo luce este archivo es el siguiente:

```
tcp  6  117  SYN_SENT  src=192.168.1.6  dst=192.168.1.9  sport=32775  dport=22
[UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22  dport=32775 use=2
```

Lo primero que aparece es el protocolo al que pertenece el paquete, en este caso a TCP. Luego le sigue la misma información pero codificada a valor decimal. Le sigue un valor que indica cuánto tiempo va a vivir esta entrada (117 segundos), la cual va a decrecer de forma regular. Luego, se presenta el estado real (dentro del protocolo) en que se encuentra la entrada (estado SYN_SENT en este caso). El valor interno del estado es significativamente diferente a los usados externamente con iptables (NEW, ESTABLISHED, etc.). El valor SYN_SENT indica que en la conexión solo se ha enviado un paquete TCP SYN en una sola dirección. Posteriormente, le sigue la dirección

IP fuente, destino, puerto fuente, puerto destino. Luego, una palabra clave indica que no hubo tráfico de respuesta para esta conexión (UNREPLIED). Por último, se muestra la información que se espera en el paquete de respuesta.

Las entradas del seguimiento de conexión pueden tomar una gran variedad de valores distintos y todos estos se encuentran especificados en los archivos de cabecera de Conntrack que se encuentran en "*linux/include/netfilter-ipv4/ip_conntrack*.h*". Estos valores dependerán del protocolo que se use por encima del protocolo IP.

Cuando una conexión tiene tráfico en ambas direcciones, la bandera [UNREPLIED] de la entrada Conntrack será eliminada y reemplazada por [ASSURED]. Esta bandera indicará que la conexión se encuentra asegurada y que no será borrada si se alcanza el máximo número de conexiones seguidas. La cantidad de conexiones que se pueden seguir está indicada en una variable que puede establecerse mediante las funciones ip-sysctl del kernel. El número por defecto dependerá de la memoria RAM que se posee, por ejemplo para 128 MB de RAM pueden almacenarse 8192 entradas, para 256 MB se obtendrán 16376 entradas. Esto puede ser configurado en el archivo "*/proc/sys/net/ipv4/ip_conntrack_max*".

3.3.2.2. Estados externos

Como se mencionó anteriormente, los paquetes pueden tomar varios estados distintos dentro del Kernel dependiendo del protocolo al que pertenezcan. Sin embargo, fuera del Kernel (en el espacio de usuario), solo existen cuatro estados que serán explicados a continuación:

- **NEW (Nuevo):** Este estado indica que se ha registrado el primer paquete de una conexión para un flujo determinado. Por ejemplo, un paquete SYN que es el primer paquete de una conexión TCP será registrado con este estado.

- **ESTABLISHED (Establecido):** Este estado indica que se ha registrado tráfico en ambas direcciones de una conexión y que se continuará registrando paquetes de la misma.

- **RELATED (Relacionado):** Este estado indica que la conexión se encuentra relacionada con otra ya establecida. Esto significa que debe existir previamente otra conexión en estado

ESTABLISHED, la cual creará otra conexión fuera de la principal. El módulo de Conntrack debe ser capaz de entender la relación entre conexiones para que establezca este estado. Un ejemplo claro para este estado es la comunicación del protocolo FTP, la cual establece dos canales de comunicación (una de control y otra de datos). La conexión del canal de datos será considerada como RELATED a la conexión del canal de control que se encontrará en el estado de ESTABLISHED.

- **INVALID (Invalido):** Este estado indica que el paquete no puede ser identificado o no posee ningún estado. Pueden haber muchas razones para que esto ocurra, tal como que el sistema se quede sin memoria, un mensaje de error ICMP que no corresponde a ninguna conexión conocida, o un paquete malformado por algún error o por algún atacante. Una buena práctica es establecer las reglas necesarias para que los paquetes que contengan este estado sean descartados.

Estos estados pueden ser usados mediante el match ***--state*** para identificar paquetes basados en su estado de seguimiento de conexión. Esto hace que la máquina de estados sea increíblemente sólida y eficiente para construir un firewall ya que permite elegir a que tráfico responder sin tener que habilitar todos los puertos de una computadora como debía hacerse antes de su desarrollo.

3.3.2.3. Conexiones TCP

Una conexión TCP es siempre inicializada mediante un acuerdo de tres vías (three-way handshake), el cual negocia y establece la conexión sobre la cual serán enviados los datos. La sesión completa es iniciada con un paquete SYN, luego se envía un paquete SYN/ACK de respuesta y finalmente un paquete ACK para notificar el establecimiento de la misma. Cuando estas tres vías finalizan, la conexión queda establecida y lista para empezar la transmisión de datos.

Desde el punto de vista del usuario, el seguimiento de conexión trabaja básicamente de igual manera para todos los tipos de conexión y no sigue exactamente el flujo de la conexión TCP como puede observarse en la Figura 3.3.2. Cuando detecta el primer paquete (SYN), considera que la conexión es NEW. Cuando divisa el paquete de respuesta (SYN/ACK), considera a la conexión como ESTABLISHED. Con esta implementación, se puede establecer las reglas correspondientes para permitir que los paquetes en estado NEW y ESTABLISHED abandonen la red local y que solo las

conexiones en estado ESTABLISHED puedan ingresar. Si la máquina de estados considerara únicamente al proceso de conexión entero como NEW, nunca se podría detener los inicios de conexiones externas hacia la red local ya que debería permitirse ingresar paquetes en dicho estado. En otras palabras, la máquina de estados no actúa exactamente igual al flujo de la conexión TCP para que sea posible establecer políticas de control de tráfico más rigurosas.

Desde el punto de vista del Kernel, existen una gran cantidad de estados internos que son usados para los seguimientos de conexiones TCP y siguen aproximadamente a los estados estándares especificados en el RFC 793 (Information Sciences Institute - University of Southern California, 1981).

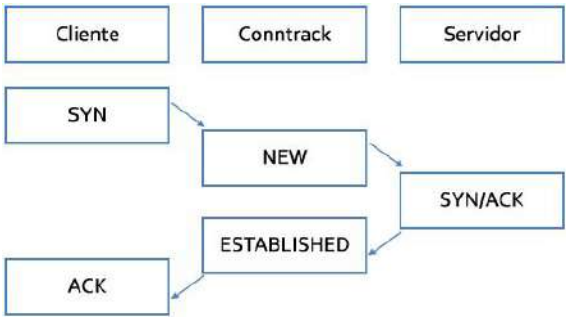


Figura 3.3.2: Estados externos para una conexión TCP.

Como se puede observar, el seguimiento de estados es bastante más simple desde el punto de vista del usuario. Sin embargo, es más complicado del lado interno del Kernel. Si se examina la base de datos del Conntrack en el establecimiento de una conexión TCP podría notarse lo siguiente:

```
tcp 6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23
[UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1
```

El primer estado (SYN_SENT) es reportado con la recepción del primer paquete SYN de la conexión, la cual aún no ha sido respondida (UNREPLIED). El siguiente estado interno será alcanzado cuando se detecte el próximo paquete en la otra dirección.

```
tcp 6 57 SYN_RECV src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23
src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1
```

La entrada anterior indica la respuesta SYN/ACK del otro lado de la conexión. El estado cambió de SYN_SENT a SYN_RECV indicando que el SYN original fue entregado correctamente y que el paquete SYN/ACK ha ingresado. Además, si bien no se hace de forma explícita, se notifica que la conexión ahora tiene tráfico en ambas direcciones mediante la remoción de la bandera UNREPLIED. El paso final será alcanzado cuando sea enviado el ACK final del acuerdo de iniciación.

```
tcp 6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23
src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1
```

Una vez recibido el paquete final del acuerdo, la conexión entrará al estado ESTABLISHED y luego del envío de unos pocos paquetes la conexión pasará al estado ASSURED.

Cuando una conexión TCP es finalizada, toma los estados vistos en la Figura 3.3.3. Como puede verse en dicha Figura, la conexión no finaliza realmente hasta que el último ACK es enviado. La Figura solo describe como la conexión es finalizada en circunstancias normales, pero también puede ser finalizada mediante el envío de un paquete RST (reset) si la conexión es denegada. En este caso, la conexión será finalizada antes de un tiempo predeterminado.

Cuando una conexión TCP ha sido finalizada, entra en el estado TIME_WAIT, el cual dura por defecto dos minutos. Este estado es utilizado para que todos los paquetes que se reciban fuera de orden puedan ser re ordenados, aún después que la conexión ya se encuentra finalizada.

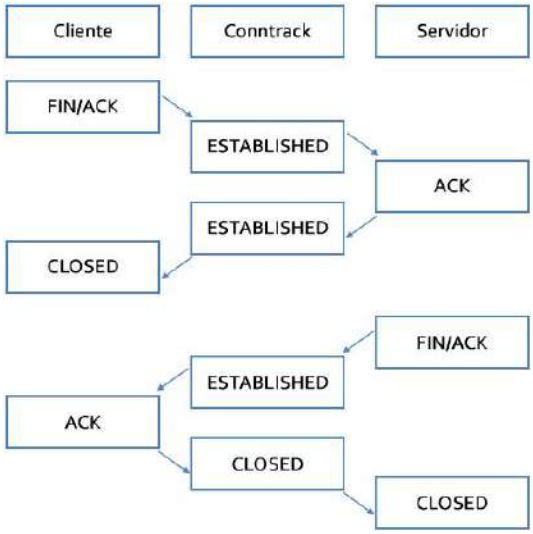


Figura 3.3.3: Estados externos al cierre de una conexión TCP.

Si la conexión es reiniciada por un paquete RST, el estado cambia a CLOSE. Esto significa que la conexión tiene diez segundos antes que sea cerrada definitivamente. Los paquetes RST no son notificados en ningún sentido, a diferencia del FIN/ACK, e interrumpen la conexión directamente. También, hay otros estados que pueden ser alcanzados los cuales son indicados en la siguiente tabla (Tabla 3.3.1) junto con sus tiempos de expiración. Estos valores pueden ser modificados en "`/proc/sys/net/ipv4/netfilter/ip_ct_tcp_*`".

Estado	Tiempo de expiración
NONE	30 minutos
ESTABLISHED	5 días
SYN_SENT	2 minutos
SYN_RECV	60 segundos
FIN_WAIT	2 minutos
TIME_WAIT	2 minutos
CLOSE	10 segundos
CLOSE_WAIT	12 horas
LAST_ACK	30 segundos
LISTEN	2 minutos

Tabla 3.3.1: Estados y tiempos de expiración.

3.3.2.4. Conexiones UDP

Las conexiones UDP no son en realidad conexiones y menos aún poseen estados. Esto se debe principalmente porque ellas no contienen ningún tipo de establecimiento o finalizado de conexión, y sus paquetes no poseen un orden secuencial. A diferencia de TCP, el hecho de recibir dos datagramas UDP en un orden específico no indica realmente que fueron enviados en ese orden. Sin embargo, es posible establecer estados a las conexiones dentro del Kernel. La Figura 3.3.4 muestra los estados externos que Conntrack le asigna a una conexión UDP.

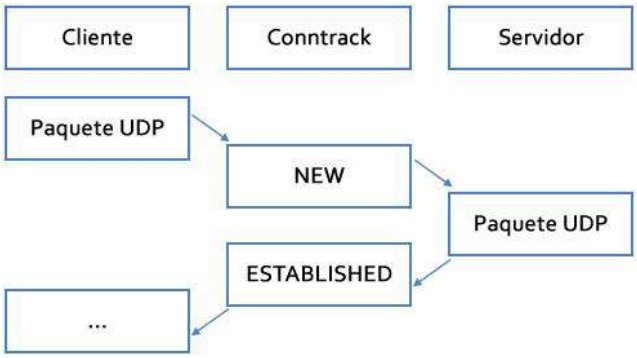


Figura 3.3.4: Estados externos para una conexión UDP.

Desde el lado del usuario, la conexión se establece de forma similar a una conexión TCP. Internamente, la información de Conntrack es un poco diferente, pero intrínsecamente el detalle es el mismo. La entrada Conntrack para un paquete UDP inicial que ha sido enviado luciría como la siguiente:

```
udp 17 30 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 [UNREPLIED]
src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1
```

El primer y segundo valor es para indicar el protocolo al que pertenece el paquete (UDP). Le sigue un valor que indica cuantos segundos va a vivir esta entrada en la base de datos. Luego, le siguen los datos que se han visto en el paquete y los que se esperan en el paquete de respuesta para el paquete inicial. Es decir, dirección fuente, dirección destino, puerto fuente y puerto destino. Entre lo enviado y lo esperado, se encuentra la bandera UNREPLIED para indicar que todavía no se ha recibido respuesta al paquete enviado. A este punto, el tiempo de expiración está establecido en treinta segundos por defecto. Cuando se reciba un paquete de respuesta se verá la siguiente entrada.

```
udp 17 180 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025
src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1
```

Al recibir una respuesta para el paquete inicial, la conexión se considera establecida (ESTABLISHED) aunque simplemente se indica mediante la remoción de la bandera UNREPLIED. Además, el tiempo de expiración para la entrada cambio de treinta segundas a ciento ochenta. Lo único que falta es la bandera ASSURED para indicar que la conexión se encuentra asegurada. Esta bandera será establecida por el módulo Conntrack luego de que haya una pequeña cantidad de tráfico sobre la conexión. Cuando esto suceda, dicha bandera aparecerá en la entrada y el tiempo de expiración se mantendrá en ciento ochenta segundos, los cuales serán reiniciados cada vez que un paquete arribe para la conexión en cuestión.

3.3.2.5. Conexiones ICMP

Los paquetes ICMP están lejos de un flujo de estados ya que ellos son utilizados únicamente para control y nunca establecen una conexión. Sin embargo, existen cuatro tipos de ICMP que generan

paquetes de respuesta y que poseen dos estados diferentes. Estos mensajes ICMP pueden tomar los estados NEW y ESTABLISHED. Los tipos ICMP en cuestión son solicitud y respuesta de eco (*Echo request*, *Echo reply*), solicitud y respuesta de marca de tiempo (*Timestamp request*, *Timestamp reply*), solicitud y respuesta de información (*Information request*, Information reply), y finalmente solicitud y respuesta de máscara de dirección (*Address mask request*, *Address mask reply*). De los cuales, Timestamp request e Information request se encuentran obsoletos, Address mask request es poca veces utilizado y Echo request/reply es el más frecuente de encontrar.

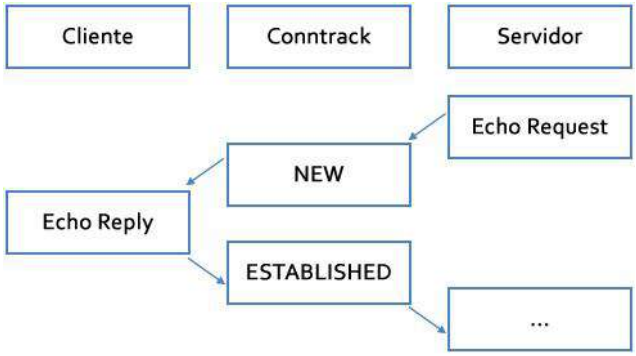


Figura 3.3.5: Estados externos para mensajes ICMP.

En la Figura 3.3.5 puede observarse los estados que toma una conexión ICMP Echo. Cuando se envía un mensaje ICMP Echo Request, Conntrack considera a la conexión como NEW. Cuando se reciba la respuesta (ICMP Echo Reply), Conntrack cambiará el estado de la conexión a ESTABLISHED. La entrada Conntrack para el primer paquete será parecida a la siguiente:

```
icmp 1 25 src=192.168.1.6 dst=192.168.1.10 type=8 code=0 id=33029 [UNREPLIED]
src=192.168.1.10 dst=192.168.1.6 type=0 code=0 id=33029 use=1
```

Al igual que en TCP y UDP, se observa el nombre y número de protocolo del paquete, la dirección fuente y destino, el puerto fuente y destino tanto del paquete enviado como el del paquete esperado. La diferencia que puede encontrarse es que esta entrada posee tres campos diferentes llamados type, code e id. Estos campos no son especiales de ningún modo, simplemente pertenecen al encabezado ICMP e indican el tipo ICMP, el código ICMP y el identificador del paquete respectivamente. Cada paquete ICMP tiene un ID que es establecido cuando es enviado, y la respuesta ICMP a este paquete contendrá el mismo ID para indicar que es la respuesta a ese mensaje y no a otro. El siguiente campo es nuevamente la bandera UNREPLIED la cual, al igual que

con los otros protocolos, indica que solo se ha detectado tráfico en una sola dirección. Cuando la respuesta arribe, la entrada lucirá con la direcciones y puertos invertidos (el origen será destino y viceversa). Los valores de los campos *type* y *code* serán corregidos a los que correspondan para la respuesta ICMP, y el campo ID se mantendrá igual.

El paquete de respuesta (Reply) será considerado como ESTABLISHED. Sin embargo, por el hecho de que se trata del protocolo ICMP puede darse por hecho que no habrá más tráfico en esta misma conexión. Por este motivo, la entrada del Conntrack será destruida una vez que la respuesta haya atravesado toda la estructura de Netfilter. En cada uno de los tipos ICMP mencionados anteriormente, la solicitud es considerada como NEW y la respuesta como ESTABLISHED.

Las solicitudes ICMP tienen un tiempo de expiración de treinta segundos por defecto, que puede modificarse en el archivo "*/proc/sys/net/ipv4/netfilter/ip_ct_icmp_timeout*".

Otra parte importante de ICMP es que puede ser utilizado para indicarle a los hosts qué es lo que sucedió con una conexión TCP o UDP específica o con un intento de conexión. Por esta razón, muy a menudo las respuestas ICMP son consideradas como RELATED a una conexión original. Un ejemplo es cuando el host o la red son inalcanzables. Si los intentos de conexión no son exitosos, esto debería ser indicado mediante una respuesta por el otro host, pero si éste (o la red) está caído debería ser indicado por el último router que intentó alcanzar al sitio en cuestión. Como se mencionó, en un caso como este, la respuesta ICMP será considerada como RELATED.

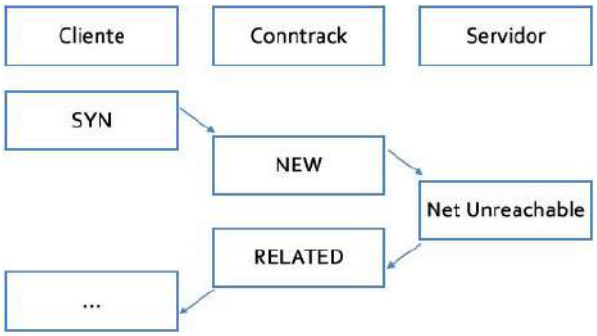


Figura 3.3.6: Estados externos para una conexión fallida.

La Figura 3.3.6 ilustra el caso descripto. Se envía un mensaje SYN para establecer una conexión TCP. Dicho paquete será considerado por Conntrack como NEW. Sin embargo, la red al que el

paquete intenta alcanzar es inalcanzable, entonces el router retorna un paquete de error ICMP (ICMP Network Unreachable). Este paquete de error será considerado como RELATED y el host seguramente abortará la conexión. Mientras tanto, Conntrack eliminará la entrada en su base de datos ya que conoce que este mensaje ha sido de error. Lo mismo ocurrirá en el caso de que una conexión UDP falle.

3.3.2.6. Conexiones por defecto

En algunos casos, la máquina de estados no conoce cómo manejar a un protocolo específico ya que no conoce cómo éste trabaja. En estos casos, trabaja en base a un comportamiento por defecto, el cual es utilizado para protocolos como por ejemplo NETBLT, MUX, y EGP. Este comportamiento es muy similar al seguimiento que hace para una conexión UDP. El primer paquete es considerado como NEW, y el tráfico de respuesta y los siguientes paquetes son considerados como ESTABLISHED.

Cuando se utiliza el comportamiento por defecto, todos los paquetes alcanzan el mismo tiempo de expiración que puede ser configurado en "*/proc/sys/net/ipv4/netfilter/ip_ct_generic_timeout*". Este tiempo de expiración está configurado por defecto en seiscientos segundos, o diez minutos. Este valor debería necesitar cambios dependiendo del tráfico que se esté intentando transmitir sobre un enlace que utilice el comportamiento por defecto. Esencialmente, si se trata de tráfico entre satélites o de ese estilo, los cuales puede tomar un largo tiempo en llegar.

3.3.2.7. Protocolos complejos

Algunos protocolos son más difíciles que otros, es decir, algunos protocolos hacen que la tarea del seguimiento de conexión sea más complicada de realizar correctamente. Algunos ejemplos de esto son ICQ, IRC y FTP. Cada uno de estos protocolos lleva su información dentro de su payload y por lo tanto se requiere de ayudantes especiales (connection tracking helpers) para funcionar correctamente.

El protocolo FTP primero abre una conexión llamada sesión de control FTP. Cuando se emiten comandos a través de esta sesión, otros puertos son abiertos para llevar el resto de los datos relacionados a ese comando específico. Estas conexiones pueden realizarse de dos maneras, de forma activa o pasiva. Cuando la conexión es realizada de forma activa, el cliente FTP envía al

servidor el puerto y dirección IP al cual debe conectarse. Después de esto, el cliente abre el puerto y el servidor se conecta a ese mediante su propio puerto 20 (conocido como FTP-Data) y envía los datos por este medio.

El problema surge en que Conntrack no conoce acerca de estas conexiones extras, ya que son negociadas dentro del payload del protocolo. Por este motivo, si se está usando Conntrack para configurar un firewall, no será capaz de saber que debe permitir al servidor conectarse con el cliente por estos puertos específicos.

La solución a este problema es agregar un ayudante especial al módulo de seguimiento de conexiones, el cual buscará a través de los datos en la conexión de control por una sintaxis y datos específicos. Cuando encuentre la información correcta, agregará la entrada correspondiente como RELATED y el servidor será capaz de realizar la conexión. Está secuencia es mostrada en la Figura 3.3.7.

El modo FTP pasivo trabaja en el sentido opuesto. El cliente le indica al servidor que quiere unos datos específicos, con lo cual el servidor le responderá con la dirección y el puerto al que debe conectarse. El cliente recibirá esta respuesta y se conectará a dicho puerto desde su propio puerto 20, y obtendrá los datos en cuestión.

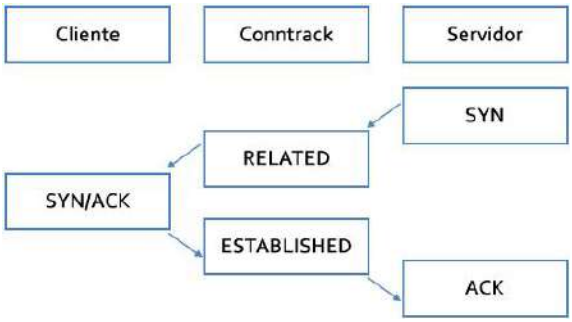


Figura 3.3.7: Estados en el establecimiento del canal de datos en FTP modo pasivo.

3.4. Control de Tráfico

Control de tráfico es el nombre que se le asigna a un conjunto de sistemas de encolamiento y mecanismos a través de los cuales los paquetes son recibidos y transmitidos en un router. Esto incluye decidir qué paquetes aceptar, a qué tasa en la interfaz de entrada y determinar qué

paquetes transmitir, en qué orden y a qué tasa por la interfaz de salida.

En la gran mayoría de los casos, el control de tráfico consiste de una única cola que recolecta todos los paquetes entrantes y los despacha tan rápido como el hardware se lo permita. A este tipo de disciplina de cola se la conoce como FIFO.

El control de tráfico es un conjunto de herramientas que le permiten al usuario tener un mejor control sobre las colas y los mecanismos de encolamiento de los dispositivos de red. El poder de reordenar los paquetes y flujos de tráfico con estas herramientas es inmenso y complicado, sin embargo, éstas no sustituyen a un ancho de bando adecuado.

El término QoS (Quality of Services) muy a menudo es utilizado como un sinónimo de control de tráfico.

3.4.1. Elementos del control de tráfico

Los mecanismos tradicionales de control de tráfico son:

- **Shaping:** Es el mecanismo por el cual los paquetes son retrasados antes de su transmisión en la cola de salida para lograr la tasa de salida deseada.

- **Shapers:** Los adaptadores intentan limitar o distribuir el tráfico para lograr o no exceder una tasa configurada (generalmente medida en paquetes por segundo o bits/bytes por segundo). Como efecto secundario, los adaptadores pueden “suavizar” ráfagas de tráfico. Una de las ventajas de shaping es la habilidad de controlar la latencia de paquetes. El mecanismo subyacente para shaping es generalmente un mecanismo token and bucket.

- **Scheduling:** Es el mecanismo por el cual los paquetes son re-ordenados entre la cola de entrada y de salida. La planificación más común utilizada es la FIFO (*first-in, first-out*). Desde un punto de vista más general, cualquier conjunto de mecanismos de control de tráfico puede ser considerado como un planificador (*scheduler*), debido a que los paquetes son reordenados a la salida.

Existen distintos mecanismos de planificación para privilegiar distintos aspectos según las necesidades, por ejemplo, prevenir que un único flujo domine el ancho de banda, alternar la cola de salida para cada flujo, prevenir sobrecargas, etc.

- **Classifying:** Es el mecanismo por el cual los paquetes son separados para distintos tratamientos, posiblemente diferentes colas de salida. Durante el proceso de aceptación, ruteo y transmisión de un paquete, un dispositivo de red puede clasificar al mismo en diferentes formas.

  El modelo de Linux le permite a un paquete ser clasificado a través de varios clasificadores dentro de la estructura de Traffic Control junto con impositores de políticas (*policiers*).

- **Policing:** Es el mecanismo por el cual el tráfico puede ser limitado. Es utilizado frecuentemente en los bordes de la red para asegurar a los pares que no se está consumiendo más ancho de banda del asignado. El mecanismo aceptará tráfico a una tasa determinada y luego realizará una acción sobre el tráfico excedente a esta tasa. La acción más dura será descartar el tráfico, aunque en lugar de esto el tráfico podría ser re-clasificado.

- **Dropping:** Es el mecanismo por el cual los paquetes son descargados.

- **Marking:** Es el mecanismo por el cual los paquetes son alterados.

En la sección 2.4.2 estos mecanismos fueron explicados para el caso particular de Servicios diferenciados.

Estos mecanismos son implementados en Linux por medio de los siguientes componentes:

Elemento	Componente Linux
Shaping	El componente ***class*** brinda capacidades de adaptación.
Scheduling	El ***qdisc*** es el planificador en Linux.
Classifying	El objeto ***filter*** realiza la clasificación a través de la mediación de un objeto ***classifier***. De forma estricta, los clasificadores no existen fuera de ***filter***.
Policing	Un ***policer*** existe solamente como una parte del objeto ***filter***.
Dropping	Para descartar tráfico se requiere a un ***filter*** con un ***policer*** que utilice la acción "descartar".
Marking	El ***dsmarkqdisc*** es utilizado para la alteración de los paquetes.

Tabla 3.4.1: Componentes Linux de control de tráfico.

Qdisc: Es un planificador (scheduler o scheduling) también llamado disciplina de cola. Se encuentra en todas las interfaces de salida dado que es el principal bloque de construcción bajo el cual se adecúa todo el tráfico de Linux. El planificador por defecto es una disciplina FIFO.

Se pueden distinguir dos tipos de qdisc: ***qdisc con clases*** y ***qdisc sin clases***. Las qdisc con clases pueden contener, como su nombre lo indica, clases y proveen un medio al cual enlazar filtros (filters). Por otro lado, las qdisc sin clases no pueden contener ninguna clase y por lo tanto tampoco es posible enlazar filtros. Dado que estas no contienen ninguna clase de hijos, no tienen utilidad para clasificar paquetes.

Por defecto existen dos qdisc: *root* e *ingress*, que en realidad no son consideradas disciplinas de colas sino más bien ubicaciones a las cuales se le pueden enlazar estructuras de control de tráfico, para el tráfico de egreso e ingreso respectivamente.

Cada interfaz contiene ambas qdisc, la principal y más común es la *root*. Esta pude contener cualquier disciplina de cola (qdisc's) con potenciales clases y estructuras de clases. Todo el tráfico que es enviado a través de una interfaz atraviesa esta qdisc.

Por otro lado, todo el tráfico aceptado por una interfaz atraviesa la ingressqdisc. Esta qdisc tiene utilidad limitada ya que no permite crear clases hijos y existe como un objeto al cual solo se le puede enlazar un filtro. Por cuestiones prácticas, esta qdisc por lo general es utilizada para aplicar políticas para limitar la cantidad de tráfico aceptado en la interfaz.

En las secciones 3.4.2 y 3.4.3 se explicarán con mayor detalle las qdisc sin clases y con clases respectivamente.

Class: Las clases solo existen en una qdisc con clases. Estas son extremadamente flexibles y siempre pueden contener, ya sea, múltiples clases o una única clase qdisc hijo. Es importante notar que una qdisc sólo puede tener hijos de la misma clase.

Cualquier clase puede tener un número arbitrario de filtros enlazados a esta (como puede observarse en la Figura 3.4.1), lo cual permite la selección de una clase hijo o el uso de un filtro para reclasificar o descartar tráfico entrando a una clase en particular.

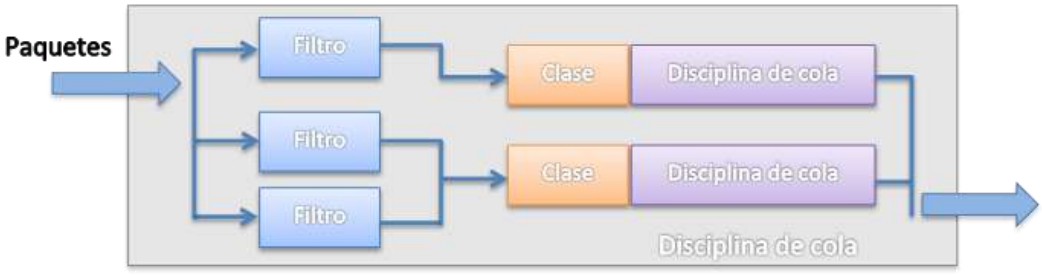


Figura 3.4.1: Disciplina de colas con clases.

Las clases que contienen una clase hijo son denominadas clases interiores o root, mientras que las que no tienen hijos son llamadas clases hojas.

Filter: Es el componente más complejo en el sistema Linux de control de tráfico. Provee un mecanismo conveniente para interconectar los distintos elementos claves del control de tráfico. El rol más simple y obvio del filtro es clasificar paquetes.

Los filtros en Linux le permiten al usuario clasificar paquetes en una cola de salida a través de varios filtros diferentes o de un único filtro (visto anteriormente en la Figura 3.4.1). Un filtro debe contener una fase de clasificación y puede contener una fase de aplicación de políticas.

Pueden ser aplicados tanto a qdiscs con clases como sin clases, sin embargo, el paquete encolado siempre entra primero a la *root* qdisc. Luego de que el filtro aplicado a la *root* qdisc es atravesado, el paquete puede ser direccionado a cualquier subclase, la cual a la vez puede tener sus propios filtros y donde el paquete puede ser nuevamente clasificado.

Classifier: Los objetos filter, que pueden ser manipulados mediante la herramienta *tc*, pueden utilizar diferentes mecanismos de clasificación, de los cuales el más común es el clasificador *u32*. Este clasificador le permite al usuario seleccionar paquetes basándose en sus atributos.

Los clasificadores son herramientas que pueden ser usadas como parte de un filtro para identificar características de un paquete o los metadatos del mismo. El objeto classifier de Linux es una analogía directa al mecanismo elemental de control de tráfico *classifying*.

Policer: Este mecanismo elemental solo es usado en el control de tráfico en Linux como parte de un filtro. Realiza acciones sobre los paquetes que superan una tasa especificada.

Aunque policing y shaping son elementos básicos del control de tráfico para limitar el ancho de banda, el objeto policer nunca retrasará el tráfico. Este sólo realiza acciones basados en algún criterio específico.

Drop: Este mecanismo básico de control de tráfico es utilizado únicamente como parte de un *policer* en el control de tráfico de Linux. Cualquier *policer* enlazado a un *filter* puede realizar una acción *drop* (descarte).

Vale aclarar que el único lugar donde un paquete puede ser explícitamente descartado en el sistema de control de tráfico de Linux es en el *policer*. Este limita la cantidad de paquetes encolados para una tasa específica o puede ser configurado para descartar todos los paquetes que coincidan con un patrón en particular.

Los paquetes pueden ser descartados por el *shaper* o el *scheduler* cuando estos no tengan espacio suficiente en sus respectivos buffers de memoria, durante ráfagas o períodos de sobrecarga.

Handle: Cada clase y qdisc con clases requiere de un identificador único dentro la estructura de control de tráfico. Este identificador único es conocido como un *handle* y está constituido por dos

miembros: un número mayor y un número menor. Estos números pueden ser asignados arbitrariamente por los usuarios de acuerdo a las siguientes reglas:

- **Mayor:** Este parámetro carece de significado para el Kernel. El usuario puede utilizar un esquema arbitrario de numeración, sin embargo todos los objetos en la estructura de control de tráfico con el mismo padre deben compartir el mismo número mayor. Los esquemas de numeración convencional comienzan con 1 para objetos directamente enlazados con la qdisc *root*.

- **Menor:** Este parámetro identifica en forma ambigua el objeto como una qdisc si su valor es 0. Cualquier otro valor identifica al objeto como una clase. Todas las clases que comparten un mismo padre deben tener valores únicos.

El *handle* es utilizado como *targets* en los parámetros *classid* y *flowid* de la sentencia de *tc filter*. Estos *handles* son identificadores externos de los objetos, solo usados por aplicaciones en el *espacio de usuarios*. El Kernel mantiene clasificadores internos para cada uno de estos objetos.

3.4.2. Disciplinas de cola sin clases

Cada una de estas disciplinas de colas puede ser utilizada como una *qdisc* primaria sobre una interfaz, o pueden ser utilizadas en una clase hoja de una *qdisc* con clases. Estas disciplinas son los principales planificadores utilizados bajo Linux.

3.4.2.1. FIFO – First-In, First-Out (pfifo y bfifo)

El algoritmo FIFO, representado en la Figura 3.4.2, forma las bases para la *qdisc* por defecto de todas las interfaces de red de Linux. Esta disciplina no realiza *shaping* ni re-ordenado de paquetes. Ésta simplemente se encarga de trasmitir los paquetes tan rápido como le sea posible después de recibirlos y encolarlos. Ésta es también la *qdisc* utilizada dentro de todas las nuevas clases creadas a menos que otra *qdisc* o clase remplace a la FIFO.

Una qdisc FIFO real debe tener un tamaño de buffer limitado para prevenir que se desborde en caso que no pueda despachar paquetes tan rápido como los recibe. Linux implementa dos qdisc FIFO básicas: una basado en bytes (*bfifo*) y otra en paquetes (*pfifo*). Sin importar del tipo de FIFO

utilizado, el tamaño del buffer es definido por el parámetro *limit*.

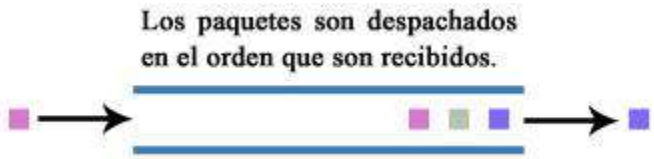


Figura 3.4.2: Disciplina de cola FIFO.

3.4.2.2. pfifo_fast, qdisc por defecto de Linux

La qdisc pfifo_fast, visualizada en la Figura 3.4.3, es la disciplina de colas por defecto para todas las interfaces en Linux. Basada en la disciplina FIFO convencional, esta qdisc además provee algún tipo de priorización. Éste provee tres bandas diferentes para separar tráfico, cada una de estas con una disciplina FIFO. El tráfico con prioridad más alta (flujos interactivos) son ubicados dentro de la banda 0 y siempre son despachados primero. Similarmente, la banda 1 es despachada antes de despachar los paquetes de la banda 2.

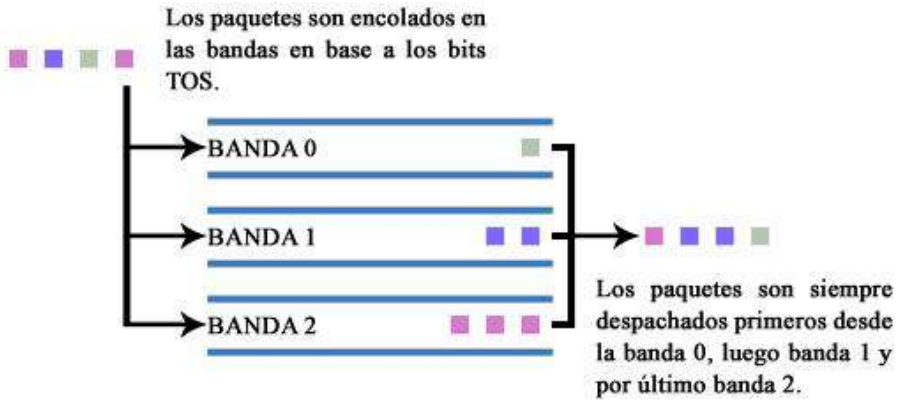


Figura 3.4.3: Disciplina de cola pfifo_fast.

El usuario no puede configurar ningún aspecto acerca de ésta qdisc. La ubicación de los paquetes en las bandas se corresponde según el valor de los bits TOS del campo ToS. En la Tabla 3.4.2 se detalla la correlación de los valores de los bits TOS y las bandas de pfifo_fast:

Bits TOS	Significado	Prioridad Linux	Banda
0	Servicio normal	0 Mejor Esfuerzo	1
1	Mínimo costo monetario (mmc)	1 Relleno	2
2	Máxima Fiabilidad (mr)	0 Mejor Esfuerzo	1
3	mmc + mr	0 Mejor Esfuerzo	1
4	Máxima transferencia (mt)	2 En Masa	2
5	mmc + mt	2 En Masa	2
6	mr + mt	2 En Masa	2
7	mmc + mr + mt	2 En Masa	2
8	Mínimo retardo (md)	6 Interactivo	0
9	mmc + md	6 Interactivo	0
10	mr + md	6 Interactivo	0
11	mmc + mr +md	6 Interactivo	0
12	mt + md	4 Int. en Masa	1
13	mmc + mt + md	4 Int. en Masa	1
14	mr + mt +md	4 Int. en Masa	1
15	mmc + mr + md	4 Int. en Masa	1

Tabla 3.4.2: Correlación entre los bits TOS y las bandas pfifo_fast.

3.4.2.3. SFQ, Stochastic Fair Queuing

La qdisc SFQ, representada por la Figura 3.4.4, intenta distribuir de forma justa las oportunidades para transmitir datos a la red entre un número arbitrario de flujos. Para lograr esto, utiliza una función hash para separar el tráfico en distintas FIFOs, los paquetes son despachados de forma round-robin entre las distintas bandas. Debido que existe la posibilidad de caer en una desigualdad en la elección de la función hash, esta función es alterada periódicamente. Esta periodicidad es establecida con el parámetro *perturb*.

Desafortunadamente, algunas aplicaciones (como Kazaa, eMule entre otros) destruyen el beneficio de este intento de lograr un encolamiento justo mediante la apertura de varias sesiones TCP. En un ambiente sin este tipo de aplicaciones, SFQ puede distribuir adecuadamente los recursos de red entre los flujos intervinientes, pero es necesario tomar otras medidas cuando existen aplicaciones que sobrecargan la red.

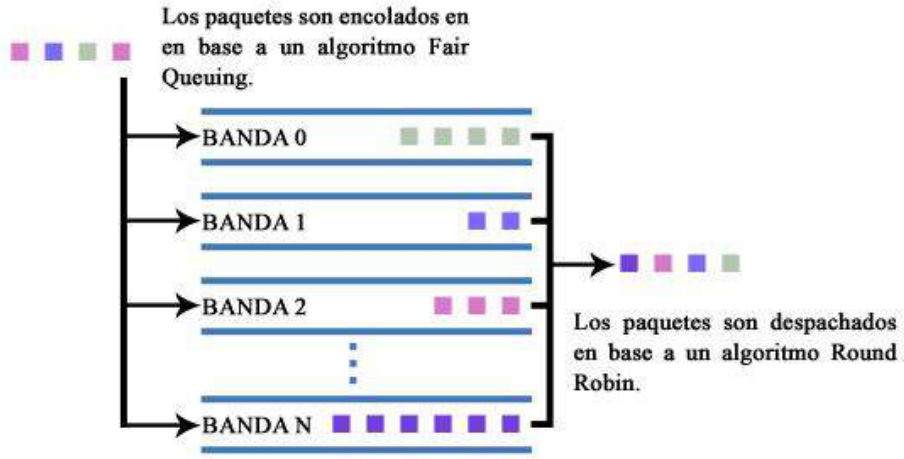


Figura 3.4.4: Disciplina de cola SFQ.

3.4.2.4. ESFQ, Extended Stochastic Fair Queuing

Conceptualmente, esta qdisc no es diferente a SFQ aunque esta le permite al usuario controlar más parámetros. Esta qdisc fue concebida para superar el problema anteriormente mencionado para SFQ. Dado que el usuario tiene mayor control y puede elegir qué algoritmo de *hashing* utilizar para distribuir el acceso a la red, es posible para el usuario final alcanzar una distribución de ancho de banda más justa.

3.4.2.5. RED, Random Early Detection

Todas las disciplinas de encolamiento que se han visto hasta ahora, e incluso TBF y HTB que se verán más adelante, basan su algoritmo de encolamiento en el conocido algoritmo de encolamiento FIFO. Esto tiene un problema serio con aquellos flujos basados en el protocolo de transporte TCP. El problema surge cuando las colas desbordan debido a ráfagas de paquetes. Es en ese momento cuando no es posible admitir más paquetes y deben ser descartados cuando están arribando al final de la cola. Es por esta razón que todas las disciplinas de colas basadas en FIFO son denominadas *Drop Tail* o *TailDrop*.

Las disciplinas *Tail Drop* distribuyen el espacio del buffer en forma injusta entre los flujos de tráfico, y también pueden llevar a una sincronización de retransmisión. Cuando esto último sucede, la súbita ráfaga de descartes debido al desbordamiento de la cola causa una ráfaga retardada de retransmisiones, que nuevamente vuelve a generar congestión. RED está orientada a

solucionar este problema.

Para ello, RED funciona con dos umbrales de tamaño de cola definidos, uno *mínimo* que establece el tamaño mínimo de la cola antes que comience el descarte de paquetes, y uno *máximo* que establece el tamaño máximo en el que intenta mantenerse. Si el promedio de paquetes calculado por el algoritmo RED supera este máximo, todos los paquetes que arriban son marcados y posteriormente descartados. Cuando el promedio de la cola se encuentra entre el máximo y el mínimo, todos los paquetes que arriban son marcados con una probabilidad de descarte, la cual consiste de una función que tiene en cuenta el tamaño promedio de la cola. La probabilidad que un paquete sea marcado para una conexión en particular es, a groso modo, proporcional a la porción de ancho de banda utilizado por esta conexión del ancho de banda total en el Gateway, es decir, que los paquetes de los flujos que consumen más cantidades del ancho de banda total tienen más probabilidades de ser descartados. De esta manera, en el caso de tener que descartar paquetes, la probabilidad de descarte de los mismos es más equitativa entre los distintos flujos.

Esta disciplina de colas es principalmente utilizada para el enrutamiento de los paquetes en las redes *backbone*.

3.4.2.6. GRED, Generic Random Early Detection

Una de las desventajas de la disciplina RED es que, a pesar de que la probabilidad de descartar los paquetes de un flujo es proporcional al ancho de banda consumido por este, nunca tiene en cuenta la prioridad de descarte que existe entre los distintos flujos que están siendo transportados a través de un Gateway.

Si se está utilizando Servicios Diferenciados (DiffServ) para determinar las características de calidad de servicio para los distintos tipos de flujos, se ha visto que, por ejemplo, utilizando Assured Forwarding existen cuatro clases AF y cada una posee tres valores posibles para determinar la precedencia de descarte de los paquetes dentro de las clases. Bajo este esquema, la disciplina RED no podría operar ya que ofrece una única cola y una única función de probabilidad de descarte para los cuatro niveles y los tres valores de precedencia. Por esta razón surge GRED como una solución a este problema.

La disciplina de cola GRED (Figura 3.4.5) está conformada por un conjunto de colas internas, las cuales cada una de ellas posee sus propios parámetros de descarte de paquetes y funcionan con una disciplina de cola RED.

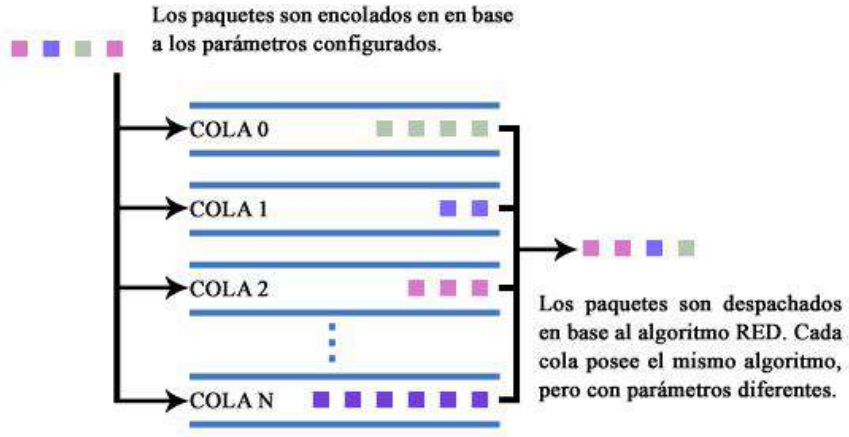


Figura 3.4.5: Disciplina de cola GRED.

DiffServ proporciona una estructura de datos llamada *sk_buff*, donde incluye un campo llamado *skb->tc_index* donde almacena el resultado de la clasificación inicial de los paquetes (valor del campo DS) que puede usarse en varios momentos del tratamiento de DiffServ. Siguiendo con el ejemplo anterior, es a través de este valor que los paquetes pueden filtrarse y ser colocados en una cola específica de acuerdo a la clase AF y precedencia de descarte al que pertenecen, y de esta manera obtener un tratamiento de salida adecuado.

3.4.2.7. TBF, Token Bucket Filter

Esta disciplina de cola está construida bajo los términos de Token y Bucket. El término *Bucket* hace referencia a un *buffer* que almacena piezas virtuales de información denominadas *Token*, que son encoladas en el *buffer* a una tasa específica denominada *token-rate*. La tasa constante en la cual se encolan los *token* en el *buffer bucket* representa la tasa máxima en que se espera que los paquetes sean despachados de la cola.

La relación que existe entre los tokens y los paquetes es de uno-a-uno, esto significa que un paquete es despachado de la cola sólo si hay al menos un token en el bucket. De otra forma, los paquetes son retenidos en la cola hasta que haya suficientes tokens.

La disciplina Token Bucket (Figura 3.4.6) adapta el tráfico que es transmitido en una interfaz. Es una solución perfecta para limitar la velocidad en que deben ser despachados los paquetes, simplemente reduce la velocidad de transmisión a una tasa especificada para el tráfico. De acuerdo a esto, pueden darse tres situaciones distintas relacionadas a la tasa en que arriban los paquetes a la cola:

1. La tasa de ingreso a la cola es igual a la *token-rate*: En esta situación no se produce ningún retardo en la salida de los paquetes por la interfaz de salida. Los paquetes arriban a la cola a la misma velocidad en que los tokens son ubicados en el bucket, por lo tanto cada paquete tiene disponible un token a tiempo.

2. La tasa de ingreso a la cola es menor a la *token-rate*: En esta situación existe una acumulación de tokens en el bucket (hasta el tamaño máximo del buffer) debido a que ingresa una menor cantidad de paquetes a la cola. En este caso, los tokens excedentes pueden ser utilizados para enviar los paquetes a una velocidad mayor a la establecida por el *token-rate*, generando cortas ráfagas de paquetes.

3. La tasa de ingreso a la cola es mayor a la *token-rate*: En esta situación existe acumulación de paquetes en el *buffer* debido a la escasez de tokens. Los paquetes son retrasados hasta que hayan tokens disponibles, si siguen arribando paquetes puede producirse la pérdida de los mismos si el *buffer* de la cola está lleno.

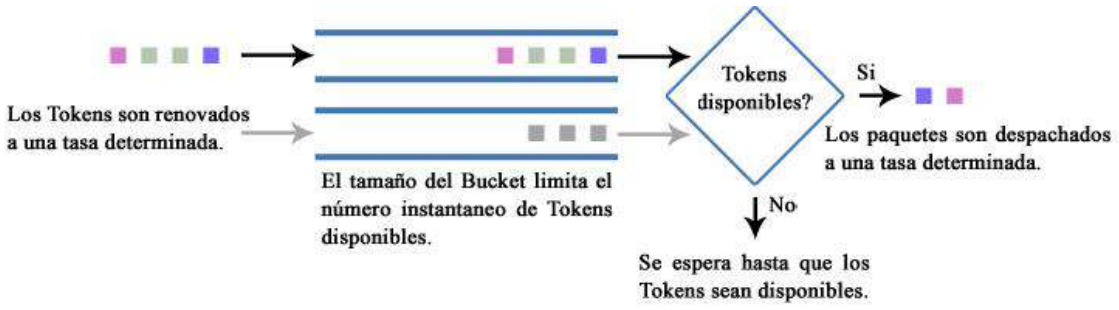


Figura 3.4.6: Disciplina de cola TBF.

3.4.3. Disciplinas de cola con clases

El control y la flexibilidad del control de tráfico de Linux pueden ser mejorados por medio del uso de disciplinas de cola con clases (classful qdisc). Estas disciplinas pueden tener filtros, permitiendo que los paquetes puedan ser dirigidos a una clase o subcola en particular.

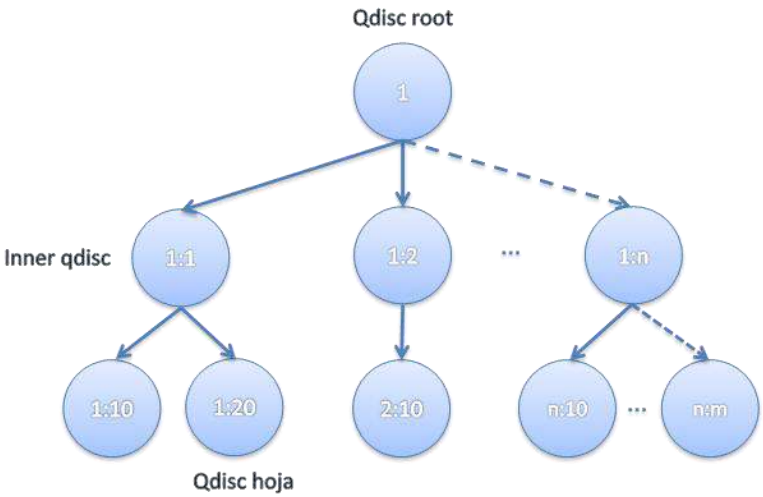


Figura 3.4.7: Disciplina de colas con clases.

La estructura de disciplinas de cola cuando se utilizan clases es análoga a la estructura de árbol comúnmente utilizada en informática (Figura 3.4.7). La raíz del árbol está representada por la qdisc root. Todas las clases directamente enlazadas a ésta son conocidas como clases root, y son generalmente clases internas (inner qdisc). Las clases finales, es decir, las que no tienen otras clases enlazadas a ellas son conocidas como clases hojas. Además del uso del lenguaje figurativo para representar la estructura de clases como un árbol, también es común el uso del lenguaje de relaciones de familia (clases padre, clases hijas, etc.).

3.4.3.1. HTB, Hierarchical Token Bucket

Esta disciplina de cola utiliza los conceptos de *token* y *bucket* (vistos en *TBF*) junto con clases y filtros que permiten un control más complejo y granular del tráfico. Además, posee un modelo complejo de préstamo de tokens, que será explicado más adelante, con el cual puede realizar una gran variedad de técnicas sofisticadas de control de tráfico. HTB permite a los usuarios definir las características de los tokens y los buckets utilizados, y anidar buckets de una manera arbitraria

dando como resultado que el control de tráfico sea más complejo.

Una de las maneras más fáciles de utilizar HTB es por medio del *shaping*, este ocurre únicamente en las clases *hojas* dado que las clases internas o *root* sólo existen, en este caso, para sugerir en qué manera el modelo de préstamo debería distribuir los tokens disponibles.

Clase	Estado de la clase	Estado HTB interno	Acción a tomar
Hoja	La tasa no fue alcanzada.	HTB_CAN_SEND	La clase hoja despachará paquetes encolados mientras haya *tokens* disponibles.
Hoja	La tasa fue alcanzada pero no se alcanzó el tope.	HTB_MAY_BORROW	Las clases hojas intentarán pedir prestadas *tokens* a sus clases padre. Si hay *tokens* disponibles, estos serán lentamente incrementados en un *quantum* y las clases hojas despacharán paquetes mientras haya *tokens* disponibles.
Hoja	El tope fue superado.	HTB_CANT_SEND	No se despacharán paquetes. Esto causará un retardo de los paquetes e incrementará la latencia para lograr la tasa requerida.
Root, interna	La tasa no fue alcanzada.	HTB_CAN_SEND	Las clases intermedias les prestarán *tokens* a sus hijos.
Root, interna	La tasa fue alcanzada pero no se alcanzó el tope.	HTB_MAY_BORROW	Las clases intermedias intentarán pedir prestados *tokens* a las clases padre, prestándoselas a los hijos que estén compitiendo por el incremento de *quantums*.
Root, interna	El tope fue superado.	HTB_CANT_SEND	Las clases intermedias no intentarán pedir prestado *tokens* a sus padres y no prestarán *tokens* a sus hijos.

Tabla 3.4.3: Estados y comportamiento de HTB.

Una parte fundamental de la disciplina HTB es el mecanismo de préstamo. Las *claseshijas* piden prestadas *tokens* a sus *padres* una vez que ellas exceden una tasa. La *clase hija* continuará intentando pedir *tokens* hasta que alcance un determinado tope (*ceil*). Llegado a este punto, comenzará a encolar los paquetes para transmitirlos cuando tenga nuevamente *tokens*

disponibles. La Tabla 3.4.3 identifica los posibles estados y comportamientos del mecanismo de préstamos.

La Figura 3.4.8 ilustra la manera en que los *tokens* son prestados y la manera en que éstos son cargados a sus padres. Para que el modelo de préstamo funcione, cada clase debe llevar una cuenta exacta del número de *tokens* usados por sí misma y por todos sus hijos. Por esta razón, cualquier *token* utilizado en un hijo o clase hoja debe ser cargado a cada clase padre hasta alcanzar la clase *root*.

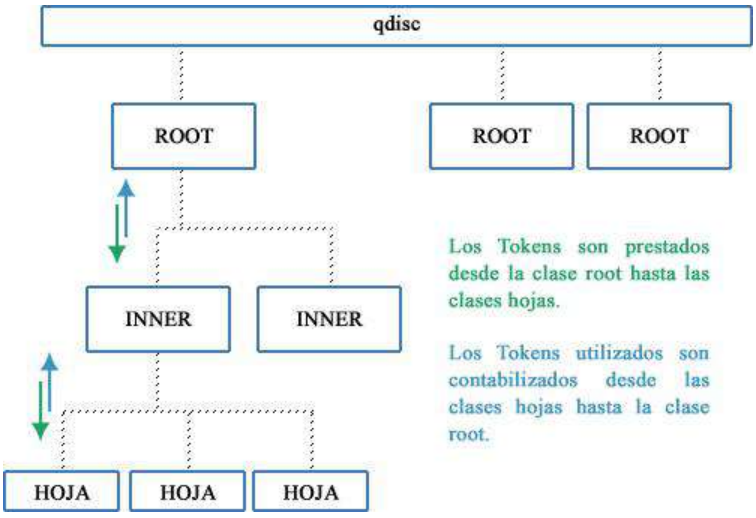


Figura 3.4.8: Mecanismo de préstamo HTB.

Cualquier clase que desea pedir prestado un *token* debe hacerlo a su clase padre, el cuál si a su vez alcanzó su tasa deberá también pedir prestado *tokens* a su clase padre. Este proceso continuará hasta que se localicen *tokens* disponibles o se alcance a la clase *root*. De esta manera, el préstamo de *tokens* fluye hacia las *clases hojas* mientras que la carga de uso de tokens fluye hacia la *clase root*.

3.4.3.2. PRIO, PriorityScheduler

Esta disciplina de cola con clases no realiza *shaping* del tráfico y puede ser vista como una *pfifo_fast* con clases. Mientras que *pfifo_fast* posee tres bandas para encolar los paquetes, PRIO posee solamente tres clases.

Los paquetes son clasificados según los filtros configurados y encolados a las clases correspondientes. La manera en que los paquetes son despachados es liberando los paquetes pertenecientes a la primer clase. Cuando ésta se encuentra vacía se despachan los paquetes de la siguiente clase.

La diferencia principal de PRIO con *pfifo_fast* es que esta última utiliza solo colas FIFO y encola los paquetes en cada banda de acuerdo a los bits TOS. En cambio, el primero puede utilizar cualquier otro tipo de disciplina (SQF, TBF, RED, etc.) para despachar paquetes en cada una de las tres clases y el encolamiento de los paquetes en cada una de ellas lo realiza según los filtros configurados y no necesariamente según los bits TOS.

3.4.3.3. DSMARK

DSMARK ofrece todas las capacidades necesarias para utilizar Servicios Diferenciados. Esta disciplina, en comparación con otras, no adapta, controla o limita el tráfico y no prioriza, retarda, reordena o descarta paquetes.

Como se mencionó en la sección 3.4.2.6, DS posee una estructura de datos llamada *sk_buff*, que incluye un campo llamado *skb->tc_index* donde se almacena el resultado de la clasificación inicial de los paquetes. La disciplina DSMARK toma el valor del campo DS de los paquetes que arriban a la interfaz para actualizar su valor y modificar el valor de *tc_index*. Los paquetes son marcados en el momento en el que son despachados con un valor entero definido para cada clase cuando se configura la disciplina de cola.

Es posible crear *n* clases que forman la disciplina, numeradas desde 1 a n-1, generando una tabla interna necesaria para implementar el comportamiento de la cola. Este parámetro también es denominado *índice*.

La estructura de la tabla interna generada por la disciplina está compuesta por índice, máscara, valor. Los valores de *máscara* y *valor* son introducidos en la construcción de la disciplina, se utilizan para generar la nueva marca de los paquetes. La marca resulta de la siguiente operación:

```
Nuevo valor DS = (valor DS actual ANDMáscara) ORvalor
```

Los paquetes cuando ingresan a la disciplina, son filtrados y asignados a alguno de los índices de la tabla interna. Con los valores mencionados, se actualiza el valor del campo DS aplicando operadores lógicos (AND y OR). Si la marca del paquete no coincide con ninguno de los filtros utilizados para la correlación de flujos con las clases creadas, se utiliza un valor de *índice* por defecto establecido en el momento de crear la disciplina DSMARK.

Considerando la Tabla 3.4.4 se presentarán dos ejemplos del funcionamiento de DSMARK, pero primero se explicará su contenido. La primera columna (*Clase*) indica las clases de servicio disponibles para Servicios diferenciados. La columna DP hace referencia al valor de precedencia de descarte de los paquetes para las clases de Assured Forwarding (AF). La columna DSCP contiene los valores de *codepoint* de los primeros seis bits del campo DS. La columna b-DSCP y x-DSCP contienen el valor binario y hexadecimal respectivamente de los bits del campo DSCP con dos ceros a la izquierda. Las últimas dos columnas representan los valores en binario y hexadecimal del campo DS teniendo el valor de los bits de ECN siempre en 00.

Clase	DP	DSCP	b-DSCP	x-DSCP	b-DS	x-DS
AF1	1	001010	0000-1010	0xa	0010-1000	0x28
	2	001100	0000-1100	0xc	0011-0000	0x30
	3	001110	0000-1110	0xe	0011-1000	0x38
AF2	1	010010	0001-0010	0x12	0100-1000	0x48
	2	010100	0001-0100	0x14	0101-0000	0x50
	3	010110	0001-0110	0x16	0101-1000	0x58
AF3	1	011010	0001-1010	0x1a	0110-1000	0x68
	2	011100	0001-1100	0x1c	0111-0000	0x70
	3	011110	0001-1110	0x1e	0111-1000	0x78
AF4	1	100010	0010-0010	0x22	1000-1000	0x88
	2	100100	0010-0100	0x24	1001-0000	0x90
	3	100110	0010-0110	0x26	1001-1000	0x98
EF	-	101110	0010-1110	0x2e	1011-1000	0xb8

Tabla 3.4.4: Valores del campo DS y Codepoints de AF y EF en DiffServ.

Con estos valores si se desea que cualquier paquete entrante sea clasificado como un DS AF11, la máscara a utilizar debería ser 0x0 y el *valor* 0x28. Este ejemplo se ve en la Figura 3.4.9.

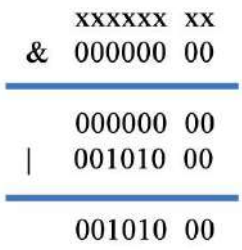


Figura 3.4.9: Marcado de paquetes con DSMARK.

Si además deseamos que se conserven los valores ECN de dichos paquetes, la máscara en lugar de ser 0x0 debería ser 0x3. Ejemplo en la Figura 3.4.10.

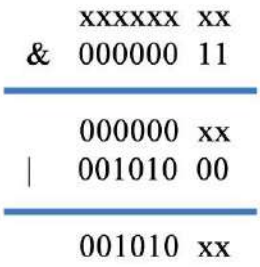


Figura 3.4.10: Marcado de paquetes con DSMARK manteniendo ECN.

3.4.4. Filtros para la clasificación de paquetes

Como se mencionó anteriormente, las disciplinas con clases están relacionadas con un filtro el cuál se encarga de clasificar los paquetes y determinar a cuál clase o sub-cola deben ser dirigidos.

A la hora de seleccionar un filtro es posible encontrarse con múltiples tipos, algunos filtros incluso son específicos para trabajar con una disciplina determinada. Entre los filtros que se pueden utilizar se encuentran:

- **Fw:** Filtra los paquetes en base a como un firewall los ha marcado.

- **u32:** Filtra paquetes en base al contenido de los campos de los encabezados del mismo. Lee los campos en palabras de 32 bits.

- **route:** Filtra los paquetes en base a la decisión de ruteo de los paquetes.

- **rsvp, rsvp6:** Filtra los paquetes en base a los protocolos de reservación de recursos RSVP.

- **tcindex:** Filtra los paquetes en base al marcado realizado por la disciplina de cola DSMARK.

Los siguientes argumentos son comunes para los clasificadores en general:

1. **protocol:** El protocolo que aceptará este clasificador. Por lo general se utiliza para clasificar paquetes IP.

2. **parent:** Es la clase al que estará asociado este clasificador. Debe ser de una clase ya existente.

3. **prio:** Indica la prioridad de este clasificador. A menor valor mayor importancia, es decir, se comprobarán primero.

4. **handle:** Tiene distintos significados para los distintos clasificadores.

A continuación se detallarán algunos de los filtros que pueden utilizarse para clasificar los paquetes.

3.4.4.1. Clasificador u32

Este filtro permite encontrar cualquier campo de bits dentro del paquete, tanto en el encabezado IP como en el encabezado TCP del mismo, por lo cual es el filtro más potente dentro del control de tráfico de Linux. Si bien trabaja de una manera simple, es el más difícil y complejo de utilizar.

Su funcionamiento hace uso de tablas hash, haciendo que este clasificador sea eficiente aun cuando se dispone de una gran cantidad de reglas de filtrado.

El filtro u32 está compuesto de selectores y acciones. Los selectores son comparados con las palabras de 32 bits que son extraídas del paquete hasta que se encuentra la primera coincidencia. Una vez encontrada una coincidencia se lleva a cabo una acción asociada a dicho selector. Por lo

general, la acción más sencilla es dirigir el paquete a una clase en particular.

El selector u32 es el siguiente:

```
match u32 PATRON MASCARA at [OFFSET | nexthdr+OFFSET]
```

- match: Indica el inicio del selector.
- u32: Tipo de clasificador o filtro utilizado.
- PATRON: El valor a buscar dentro del encabezado del paquete.
- MASCARA: Es la máscara de comparación que se aplica sobre el paquete, son los bits exactos (de la palabra de 32 bits) que se deben comparar. La comparación debe dar como resultado el PATRON buscado.
- OFFSET: Posición dentro del encabezado en la cual se hará la comparación.
- nexthdr+OFFSET: Si el parámetro no se indica, la búsqueda se realiza en el encabezado de la capa de red del paquete (protocolo IP). Si se indica, la búsqueda se realizará en el encabezado de la capa de transporte (TCP, UDP, etc.). OFFSET tiene el mismo significado que el explicado recientemente.

3.4.4.2. Clasificador route

Este tipo de filtro funciona en base al resultado de la tabla de ruteos al momento de seleccionar la ruta por la cual los paquetes deben ser transmitidos.

Las rutas son marcadas con un valor entero llamado REALM en las tablas de ruteo con el comando *ip route* de la siguiente manera:

```
ip route add HOST/RED via GATEWAY dev INTERFAZ realm REALM
```

De esta manera, los paquetes que son enviados a través de estas rutas también son marcados con el valor *realm* de dicha ruta. El filtro toma esta información para determinar hacia qué disciplina de cola o clase debe enviar los paquetes. El selector route es el siguiente:

```
route [to | from] REALM
```

El parámetro *to* es utilizado para seleccionar los paquetes que tienen como destino el/la HOST/RED asociado/a con valor REALM. El parámetro *from* es utilizado para seleccionar los paquetes que tienen como origen el/la HOST/RED asociado/a con el valor REALM.

3.4.4.3. Clasificador tcindex

El clasificador *tcindex* fue diseñado específicamente para ser utilizado con Servicios Diferenciados en la arquitectura de Linux. Este filtro trabaja sobre la estructura de buffer *sk_buff* y debe ser usado únicamente por disciplinas de colas que son capaces de reconocer el contenido del campo *skb->tc_index,* como GRED y DSMARK.

Este filtro funciona con dos valores enteros llamados *mask* y *shift* definidos en un filtro principal. La operación que se realiza sobre el valor de skb->tc_index es la siguiente:

```
skb->tc_index &mask >> shift
```

La operación anterior corre *shift* posiciones a la derecha los bits de skb->tc_index que concuerden con la máscara *mask*. El valor devuelto por la operación es convertido a valor decimal y será controlado posteriormente por otros filtros tcindex que enviarán los paquetes que posean este valor DS a la disciplina correspondiente. El filtro principal se utiliza de la siguiente manera:

```
tcindex mask MASK shift SHIFT pass_on
```

El parámetro *pass_on* se utiliza para que los paquetes que no concuerden con la máscara definida sean verificados por el siguiente filtro disponible. Los filtros tcindex posteriores son definidos de la forma:

```
handle HANDLE tcindex
```

El valor HANDLE debe ser un valor entero, el cual se corresponde con algún valor posible del campo DS. De esta manera, el valor devuelto por el filtro principal al clasificar un paquete es comparado con los *handle* de los demás filtros *tcindex* definidos hasta que exista una coincidencia.

3.4.5. Comando tc

Muchas distribuciones de GNU/Linux proveen soporte para el control de tráfico en forma modular o monolítica. Los Kernels personalizados podrían no proveer soporte para las características requeridas. En tal caso, es necesario que el Kernel sea recompilado agregando las opciones necesarias mencionadas en la siguiente lista:

- CONFIG_NET_SCHED=y
- CONFIG_NET_SCH_CBQ=m
- CONFIG_NET_SCH_HTB=m
- CONFIG_NET_SCH_CSZ=m
- CONFIG_NET_SCH_PRIO=m
- CONFIG_NET_SCH_RED=m
- CONFIG_NET_SCH_SFQ=m
- CONFIG_NET_SCH_TEQL=m
- CONFIG_NET_SCH_TBF=m
- CONFIG_NET_SCH_GRED=m
- CONFIG_NET_SCH_DSMARK=m
- CONFIG_NET_SCH_INGRESS=m
- CONFIG_NET_QOS=y
- CONFIG_NET_ESTIMATOR=y
- CONFIG_NET_CLS=y
- CONFIG_NET_CLS_TCINDEX=m
- CONFIG_NET_CLS_ROUTE4=m
- CONFIG_NET_CLS_ROUTE=y
- CONFIG_NET_CLS_FW=m
- CONFIG_NET_CLS_U32=m
- CONFIG_NET_CLS_RSVP=m
- CONFIG_NET_CLS_RSVP6=m
- CONFIG_NET_CLS_POLICE=y

Esta lista de opciones puede diferir entre las distintas versiones de Kernel que se esté intentando compilar. Aquellas opciones instaladas como módulos deben ser cargados al Kernel con el comando *modprobe* antes de utilizarlos.

Como se mencionó al comienzo de este capítulo, *iproute2* ofrece un conjunto de herramientas útiles que permite manipular las estructuras del Kernel para la configuración de redes IP. Para la creación y manipulación de estructuras de control de tráfico se utiliza la herramienta *tc* de *iproute2*.

La herramienta tc permite realizar toda la configuración necesaria para llevar acabo el control de tráfico. Como primer argumento en su sintaxis puede ser uno de los tres componentes de Linux para el control de tráfico (*qdisc*, *class* o *filter*), y cada uno de estos componentes poseen diferentes opciones para su creación. A continuación se presentarán ejemplos de la creación de cada uno de estos componentes.

Para la creación de una disciplina de cola se utiliza el siguiente comando:

```
# tc qdisc add    \ [1]
>dev eth0       \ [2]
>root           \ [3]
>handle 1:0     \ [4]
>htb [5]
```

1. *add:* Agrega una disciplina de cola. Puede utilizarse *del* para eliminar una disciplina.
2. *dev:* Indica sobre cual interfaz se crea la disciplina.
3. *<tipo de qdisc>:* En este caso se utiliza *root* para indicar que es una disciplina de egreso. Para indicar la clase raíz de ingreso se utiliza *ingress*.
4. *handle:* Como se explicó en la sección 3.4.1 es un identificador definido por el usuario. Para este componente, el valor *menor* debe ser siempre cero.
5. *<disciplina de cola>:* Se indica la disciplina de cola aplicada para la qdisc. En este caso se utiliza HTB.

Luego, se creará una clase que se enlazará a la qdisc *root*.

```
# tc class add    \ [1]
>dev eth0       \ [2]
>parent 1:0     \ [3]
>classid 1:6    \ [4]
>htb\ [5]
>rate 256kbit   \ [6]
>ceil 512kbit  [7]
```

1. *add:* Agrega una clase. Puede utilizarse *del* para eliminar una clase.
2. *dev:* Indica sobre cual interfaz se crea la clase.
3. *parent:* Indica a que clase padre se enlaza la clase que se está creando a través del *handle*.
4. *classid*: Indica el *handle* que identifica a esta clase. En el valor *menor* debe ser siempre distinto de cero.
5. *<disciplina de cola>:* Las qdisc con clases requieren que las clases hijo tengan la misma disciplina de cola que su padre. Por esta razón, se indica que esta clase utiliza la disciplina HTB.
6. *rate:* Parámetro específico de HTB, establece la tasa mínima de transmisión.
7. *ceil:* Parámetro específico de HTB, establece la tasa máxima de transmisión.

Finalmente, se creará un filtro que clasificará que paquetes deben ser tratados por la clase creada en el paso anterior.

```
# tc filter add    \ [1]
>dev eth0  \ [2]
>parent 1:0\ [3]
>protocol ip\ [4]
>prio 5       \ [5]
>u32\ [6]
>match ip port 22 0xffff \ [7]
>match ip tos 0x10 0xff   \ [8]
>flowid 1:6 \ [9]
>police\ [10]
>rate 32000bps  \ [11]
>burst 10240\ [12]
>mpu 0\ [13]
>action drop/continue  [14]
```

1. *add:* Agrega un filtro. Puede utilizarse *del* para eliminar un filtro.
2. *dev:* Indica sobre cual interfaz se crea el filtro.

3. *parent:* Indica a que clase padre se enlaza el filtro que se está creando a través del *handle*.
4. *protocol:* Este parámetro es requerido, aunque parezca obvio. Se establece sobre cual protocolo se aplica el filtro.
5. *prio:* Permite establecer una preferencia entre los filtros.
6. *<clasificador>:* Es el tipo de clasificador (classifier) requerido en cada filtro.
7. *match:* Es un parámetro del clasificador, *match* se utiliza para indicar el patrón de los paquetes que serán seleccionados por el filtro. En este caso paquetes con puerto destino 22.
8. *match:* En este parámetro se seleccionan paquetes con tipo de servicio interactivos.
9. *flowid:* Específica el *handle* de la clase o qdisc al que se enviarán los paquetes seleccionados por el filtro.
10. *police:* Parámetro opcional para el filtro, indica al elemento *policier*.
11. *rate:* El *policier* realizará alguna acción por encima de esta tasa.
12. *burst:* El *policier* realizará alguna acción por encima de esta ráfaga.
13. *mpu:* La unidad mínima de "vigilancia". Para controlar todos los paquetes se debe utilizar el valor 0.
14. *action:* Indica la acción que debe realizar el *policier* si la tasa es superada. La primera palabra especifica la acción que debe tomar si se supera la tasa, la segunda palabra especifica la acción a tomar si esto no ocurre.

Mediante el ejemplo anterior se creó una política de control de tráfico en la interfaz de salida. La misma utiliza una disciplina de cola HTB que llena su bucket a una tasa de de 256 kbit/s y a una tasa tope de préstamo de tokens de 512 kbit/s. A esta política se le suma un filtro y un impositor de política que harán que todos los paquetes que tengan como destino el puerto 22 y además pertenezcan a un flujo de datos con tipo de servicio interactivo sean descartados cuando el flujo de datos supere una tasa de 32000 bps con ráfagas de más de 10240 bytes.

3.5. Resumen

La calidad suele ser apuntada como uno de los puntos altos para garantizar el éxito en cualquier emprendimiento, sea tecnológico o no, ya que son los usuarios finales quienes de manera subjetiva lo perciben. Para que esta cualidad no sea simplemente subjetiva, es importante la existencia de los TCA y SLA cuya finalidad es establecer el nivel y las características que los proveedores deben brindar en concepto de calidad.

El Framework Netfilter permite, a través de sus distintas herramientas, definir una política para el tratamiento del tráfico de un dominio. Lo primero a determinar es qué tráfico será admitido y cuál descartado. Una vez identificado el tráfico importante para el dominio, se aplicarán filtros para separar a los distintos flujos que lo atraviesan, marcarlos y encolarlos mediante una disciplina de acuerdo a los requerimientos de calidad previamente establecidos para esos flujos. Estas disciplinas harán uso de los comportamientos (Behaviors) que estarán directamente relacionados a un codepoint de acuerdo a la política de QoS que se haya implementado (PHB por defecto, AF, EF y compatibilidad con IP Precedence). Este conjunto de acciones es fundamental para el cumplimiento de los acuerdos TCA, entre los dominios, y SLA para el cumplimiento del servicio diferenciado entre el proveedor del dominio y los usuarios del mismo.

Es importante remarcar que las herramientas vistas durante este capítulo son muy importantes para la implementación de calidad de servicio, cada una de ellas cumple con un rol determinado y solo tienen una utilidad apreciable cuando son empleadas en forma conjunta.

Capítulo 4: Protocolos de señalización multimedia en VoIP

4.1. Introducción

Además de brindar Calidad de Servicio al tráfico de voz (VoIP) para que las comunicaciones no sufran retardos, perturbaciones y pérdidas al ser transmitidas en una red de conmutación de paquetes en lugar de una red de conmutación de circuitos, como es en la telefonía convencional, es necesario una gestión de recursos que asegure la optimización de la capacidad de transporte de la voz extremo a extremo mediante la utilización de protocolos de señalización.

Los protocolos de señalización surgen como consecuencia de la masividad de servicios multimedia utilizados sobre redes IP, como streams de video, audio, mensajería instantánea, etc. Posteriormente, algunos de estos protocolos debieron adaptarse a las necesidades de la transmisión de voz sobre IP, como es el caso del protocolo SIP.

Dentro de su funcionamiento, los protocolos de señalización pueden realizar varias operaciones en las que se puede encontrar:

- Login de los extremos de la comunicación en el servidor.
- Establecer, modificar y terminar una sesión entre los extremos.
- Indicar que tipo de datos serán transmitidos durante la comunicación.
- Indicar los codecs que se utilizan en una sesión.
- Transmitir la dirección de origen y destino de las comunicaciones.

No obstante, la utilización de estos protocolos de señalización hace que la cantidad de información necesaria para ser intercambiada entre los dispositivos importantes (hosts, routers, etc.), se incremente significativamente antes que pueda establecerse una sesión multimedia. Esta sobrecarga de tráfico puede ser peor si se considera el caso de acceder a contenidos multimedia de alta calidad o servicios multimedia que se han vuelto más populares, como ha sucedido en la actualidad. De esta manera, el acceso a contenidos podría tomar varios segundos, o tal vez minutos, lo cual es claramente incompatible con las expectativas de los clientes. La Internet Engineering Task Force (IETF) ha desarrollado distintos protocolos para la señalización y control de

sesiones multimedia entre los cuales se encuentran SIP (Session Initiation Protocol) y MGCP (Media Gateway Control Protocol).

Además de estos protocolos de señalización, es necesaria la utilización de otros protocolos en forma conjunta que serán los encargados de transportar los datos multimedia una vez que la sesión haya sido establecida. Por ejemplo, el protocolo SDP (*Session Description Protocol*) es a menudo solicitado por SIP para describir el contenido de los datos que serán transportados por otro protocolo como es RTP (*Real-Time Transport Protocol*) que es el verdadero protocolo encargado de transportar la información entre los extremos de la sesión establecida por SIP.

Los protocolos de señalización para VoIP más frecuentemente utilizados son H.323, SIP e IAX/IAX2 que serán tratados en este capítulo.

4.2. Protocolo H.323

H.323 es un estándar publicado por la ITU-T (International Telecommunications Union Telecommunications Sector) que define protocolos para proveer sesiones de comunicación audiovisual sobre cualquier red de conmutación de paquetes. Se encuentra dirigida al control y señalización de llamadas, control y transporte de multimedia, y control de ancho de banda para conferencias punto a punto y multipunto.

La señalización de llamadas H.323 está basado en la recomendación Q.931 del ITU-T y está adecuada a llamadas a través de redes mixtas de IP, PSTN e ISDN (Red digital de servicios integrados). Su modelo de llamadas, similar al de ISDN, facilita la introducción de telefonía IP en las redes existentes de los sistemas PBX basados en ISDN, incluyendo transiciones a PBXs basadas en IP.

En la siguiente tabla se destacan algunos de los protocolos definidos en el estándar H.323 clasificados según su propósito:

Propósito	Protocolo	Descripción
Señalización y control de llamadas	H.225.0	Protocolo de señalización de llamadas y empaquetado de flujos multimedia.
	H.225.0/RAS (Registration, Admission and Status)	Utilizado entre un extremo H.323 y un GateKeeper para proveer servicios de resolución de dirección y control de admisión.
	H.245	Describe mensajes y procedimientos usados para el intercambio de capacidades, apertura y cierre de canales lógicos de audio, video y datos, control y notificaciones.
Conferencia de datos	T.120	Conjunto de protocolos para transmisión de datos entre los extremos y está estructurado en capas tal como el modelo OSI.
Transporte multimedia	RTP	Utilizado para enviar y recibir información multimedia en tiempo real.
	RTCP (RTP de control)	Proporciona información de control asociado a los flujos de datos transmitidos mediante RTP.
Seguridad	H.235	Seguridad y encriptación para los protocolos de la familia H.323.
Servicios suplementarios	H.450.1	Funciones genéricas para el control de servicios suplementarios en H.323.
	H.450.2	Función de transferencia de llamadas.
	H.450.3	Función de desvío de llamadas.
	H.450.4	Función de llamada en espera.
	H.450.5	Función de estacionamiento y atendido de llamadas
	H.450.6	Función de llamada en segunda línea.
	H.450.7	Función de notificación de mensajes en el contestador.
	H.450.8	Servicio de notificación de nombres.
	H.450.9	Finalización de llamada.

Tabla 4.2.1: Protocolos definidos por H.323.

Además este estándar define un conjunto de codecs para la codificación de los datos multimedia transmitidos. A continuación se listan algunos codecs de video, audio y texto utilizados dentro de este estándar.

- **Audio**
  - **G.711:** Es un estándar para representar señales de audio con frecuencias de la voz humana, mediante muestras comprimidas de una señal de audio digital con una tasa de muestreo de 8000 muestras por segundo (8kHz). Proporciona un flujo de datos de 64 kbit/s.
  - **G.722:** Codifica 7 kHz el ancho de banda del audio a un flujo de datos de 48-64 kbit/s.
  - **G.723.1:** Utiliza dos tasas de codificación de voz (a un muestreo de 8 kHz) para comunicaciones multimedia que serán trasmitidas a 5.3 kbit/s y 6.3 kbit/s.
  - **G.728:** Codifica 3.4 kHz el ancho de banda del audio a un flujo de datos de 16 kbit/s.
  - **G.729:** Codifica 8 kHz el ancho de banda del audio a un flujo de datos de 8 kbit/s.
- **Video**
  - **H.261:** Fue diseñado para transmitir video a una tasa entre 4 kbit/s a 2 Mbit/s. Soporta dos tipos de resolución, la CIF (Common Interchange Format) de 352x288 pixels y QCIF (Quarter Common Interchange Format) de 176x144 pixels.
  - **H.263:** Este códec fue diseñado para videoconferencias. Trabaja a una tasa de 64 kbit/s. Soporta QCIF, CIF, 4CIF (704×576 pixels) y 16 CIF (1408×1152 pixels).
  - **H.264:** Códec de video de alta compresión, se ha vuelto muy popular en la actualizad y posee un algoritmo de predicción que permite comprimir los datos de forma más eficiente que otros codecs de video.
- **Texto**
  - **T.140:** Conocido como ToIP (Text over IP). Suponiendo un tipeo de 30 caracteres por segundo, resulta en un tráfico de datos de 2 kbit/s.

En la Figura 4.2.1 se puede observar de forma más organizada la pila de protocolos provistos por

H.323 para el soporte voz, video y otros medios de comunicación.

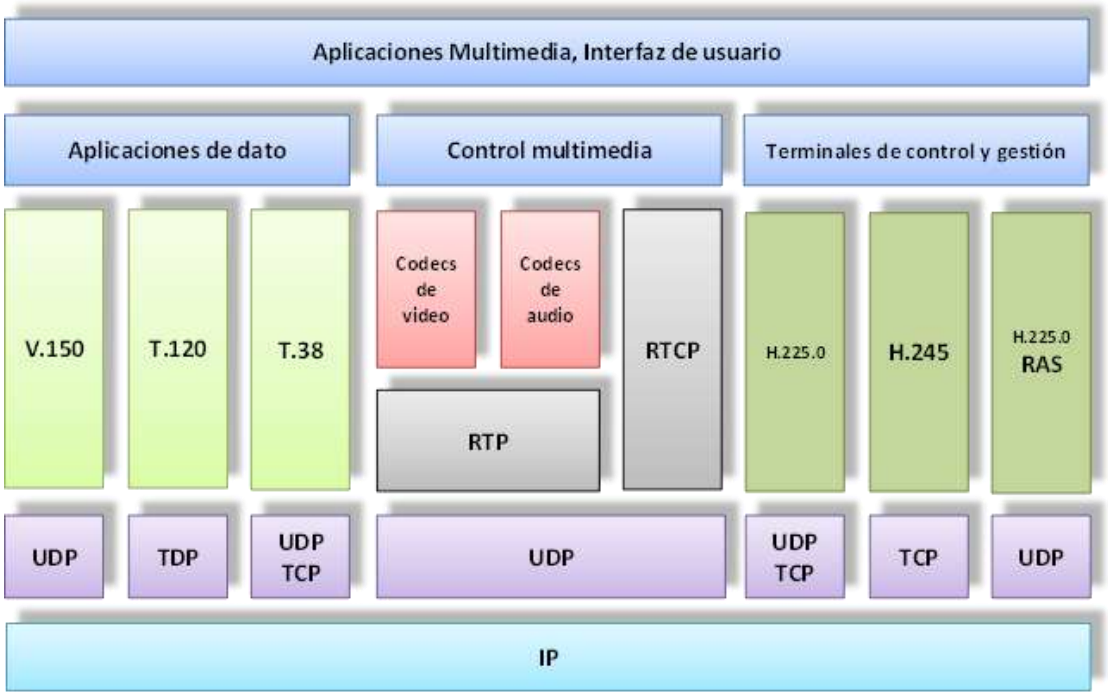


Figura 4.2.1: Pila de protocolos H.323.

4.2.1. Arquitectura del estándar H.323

Este estándar define distintos tipos de elementos de red que trabajan en conjunto para entregar capacidades de comunicación multimedia. Estos elementos son:

- Terminals (Terminales)
- Multipoint control units (Unidades de control multipunto)
- Gateways (Puertas de enlace)
- Gatekeeper (Portero)
- Border elements and peer elements (Elementos de borde y elementos pares)

4.2.1.1. Terminals

Las terminales son los elementos fundamentales de cualquier sistema H.323, ya que son los dispositivos que los usuarios normalmente utilizarán. Estos pueden existir en la forma de teléfonos IP o poderosos sistemas de videoconferencia de alta definición.

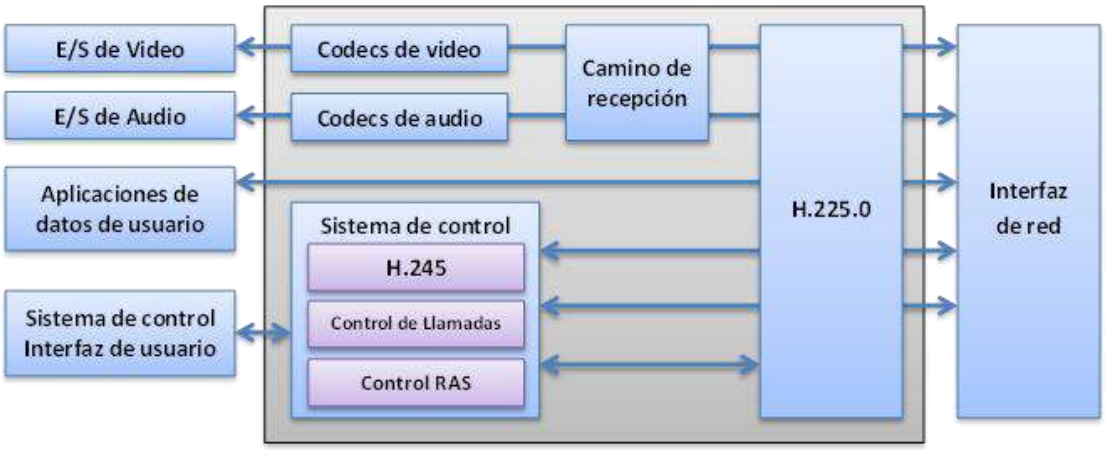


Figura 4.2.2: Terminal H.323.

Proporciona en tiempo real comunicación bidireccional con otra terminal H.323, puerta de enlace o unidad de control multipunto. El intercambio de información incluye controles, notificaciones, audio, video y datos. Una terminal debe soportar al menos transmisión de:

- Voz,
- Voz y datos,
- Voz y video,
- Voz, datos y vídeo.

4.2.1.2. Multipoint Control Units (MCU)

Es el responsable de gestionar conferencias multipunto y está compuesto de dos entidades lógicas denominadas como Multipoint Controller (MC) y Multipoint Processor (MP). El MC es el encargado de proporcionar capacidad de negociación y control de los miembros de los grupos y el MP se encarga de realizar las funciones de mezcla de medios (audio, vídeo, datos).

Algunos MCU además proveen capacidades de colaboración de datos multipunto. Lo que significa para el usuario final que este podría ver a todos los participantes de una conferencia en vez de oír solo sus voces.

4.2.1.3. Gateway

Los gateways son los dispositivos capaces de comunicar las redes H.323 con otras redes, tales como las redes PSTN o redes ISDN. Si uno de los miembros de una conversación utiliza una

terminal no H.323, entonces la comunicación deberá pasar a través del gateway para que los participantes sean capaces de comunicarse.

Las gateways pueden conectarse directamente con terminales H.323 o bien con otras gateways o terminales en otras redes y realiza las funciones de adaptación entre flujos de información y los protocolos de control de ambos entornos.

4.2.1.4. GateKeeper

Es un componente opcional de las redes H.323 que provee un número de servicios a las terminales, gateways y MCU. Estos servicios incluyen resolución de direcciones, autentificación de usuarios, autorización de llamadas, control de admisión, control de zonas, gestión de ancho de banda, gestión de llamadas, reserva de ancho de banda, servicios de directorio, etc. De las varias funciones llevadas a cabo por el gatekeeper, la más importante es la resolución de direcciones ya que esta permite a dos extremos contactarse sin tener que conocer la dirección IP del otro.

Una colección de extremos registrados en un solo gatekeeper es conocida como *zona*. Esta colección no necesariamente debe poseer una topología física asociada. Es decir, una zona podría ser enteramente lógica y definida arbitrariamente por el administrador de la red.

4.2.1.5. Border elements and peer elements

Son elementos opcionales similares a los gatekeepers, pero no gestionan a los extremos directamente y proveen algunos servicios que no están detallados en el protocolo RAS.

Los elementos bordes son entidades de señalización que se sitúan en los bordes de un dominio administrativo (colección de zonas) y se comunican con otros dominios administrativos. Esta comunicación puede incluir información de autorización de acceso de precio de llamada o datos importantes necesarios para habilitar la comunicación entre dos dominios administrativos.

Los elementos pares se encuentran dentro de un dominio administrativo y ayudan a propagar la información obtenida por los elementos bordes dentro del dominio administrativo. Esta arquitectura está dirigida para permitir desarrollos a gran escala.

4.2.2. Fases de comunicación

Mediante este estándar, la comunicación entre dos extremos (supóngase A y B) que dependen de un mismo Gatekeeper puede llevarse a cabo siguiendo las siguientes fases:

1. **Establecimiento de llamada:** El extremo A, que inicia la llamada, se contacta con el gatekeeper (GK) a través de mensajes RAS mediante ARQ para obtener la identificación del usuario que inicia la llamada. El gatekeeper puede aceptar la comunicación respondiendo con un mensaje ACF, el cual contendrá entre otras cosas la dirección IP y puerto destino de la llamada, o rechazarla mediante un mensaje ARJ. En el caso de que la llamada sea aceptada, el extremo A establece una conexión TCP con el extremo B para establecer un canal de señalización H.225.0. El extremo B, al recibir esta conexión, se contacta con el gatekeeper a través de mensajes RAS para obtener permiso para contestar la llamada. Si obtiene el permiso, a través de la conexión TCP negociará la dirección y puerto por el cual se establecerá el canal de señalización para la negociación de parámetros y control de la comunicación. La conexión TCP será finalizada ya que las negociaciones se realizarán a través del canal de señalización.

2. **Intercambio de capacidades:** Mediante una conexión TCP, se establece el canal H.245, este será utilizado para establecer los parámetros de comunicación. Estos parámetros pueden ser codecs a utilizar, número de conexiones y direcciones a utilizar, puertos, número de muestras, funcionamiento maestro-esclavo, etc, que permite establecer los canales de audio, video y datos. Esta conexión debe mantenerse mientras se intercambien información entre las terminales y permite modificar parámetros en el transcurso de la comunicación.

3. **Intercambio de información audiovisual:** En este punto, ambos extremos establecen canales de información a través de la arquitectura RTP/UDP/IP para el transporte de audio, video y datos, así como canales de control a través de la arquitectura RTCP/UDP/IP para los canales de realimentación, con el objetivo de controlar la calidad de los flujos de información recibida por el otro extremo de la comunicación.

4. **Finalización de llamada:** Para finalizar la llamada, los extremos deben informarse a través del canal H.245 mediante el envío de la primitivas de finalización de llamadas, que finalizará con el envío de la primitiva EndSessionCommand que provocará el cierre del canal. Además deberán informar al GK mediante el envío del mensaje RAS Disengage Request (DRQ) que permitirá al GK liberar recursos y proporcionar información de tarificación entre otras.

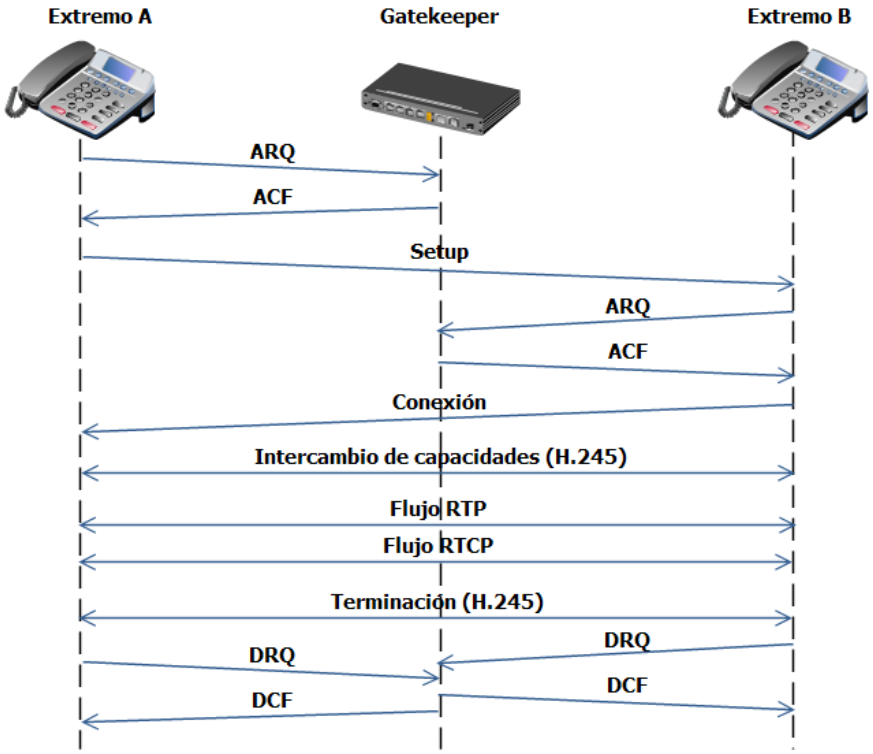


Figura 4.2.3: Conexión H.323.

4.3. Protocolo SIP

El Protocolo de Iniciación de Sesión (SIP) es un protocolo de señalización en la capa de aplicación diseñado por la IETF y especificado en el RFC 3261 (J. Rosenberg, 2002) para crear, modificar, y terminar sesiones con unos o más participantes. Estas sesiones incluyen llamadas telefónicas por Internet, distribución de datos multimedia, y conferencias multimedia. Además, puede agregar o quitar participantes o datos multimedia a sesiones ya existentes.

SIP soporta cinco fases de establecimiento y finalización multimedia:

- **Ubicación de usuario:** Asegurar que la llamada llegue a destino donde sea que el usuario se encuentre, transportando cualquier información necesaria para describir su ubicación.
- **Disponibilidad del usuario:** Determinar la disposición del invitado para participar en una comunicación.
- **Capacidades del usuario:** Determinar los medios de comunicación que serán utilizados y sus parámetros.
- **Configuración de sesión (ringing):** Establecimiento de los parámetros de sesión para ambos lados de la comunicación.
- **Gestión de sesión:** Incluye transferencia y finalización de sesiones, modificación de parámetros de sesión, e invocación de servicios.

SIP puede ser utilizado en conjunto con otros protocolos para proveer un servicio completo a los usuarios. Sin embargo, su funcionamiento básico y operación no depende de estos protocolos. En palabras más exactas, SIP no provee servicios, sino más bien provee primitivas que pueden ser utilizadas para implementar los distintos servicios. Por ejemplo, SIP puede ubicar a un usuario y entregar un objeto a su ubicación actual. Si esta primitiva es utilizada para entregar una descripción de sesión escrita en SDP, los extremos pueden negociar los parámetros de sesión. Si esta misma primitiva es utilizada para entregar una foto del participante que inicia la sesión así como también la descripción de la sesión, se podría implementar fácilmente un servicio de "Caller ID".

El protocolo SIP sólo se encarga de la señalización de la sesión. Una vez establecida esta, los datos multimedia son transmitidos en forma independiente a través de otros protocolos, como por ejemplo RTP y RTCP. Los datos multimedia viajan directamente entre los extremos de la comunicación y no necesariamente son transportados por los mismos saltos por el cual se estableció la sesión.

Dentro de las características de SIP se puede destacar que:

- Está basado en el modelo Cliente/Servidor.
- Tiene sintaxis similar a HTTP o SMTP. Los mensajes consisten de encabezados y un cuerpo de mensaje en texto que utiliza la codificación UTF-8.

- Las aplicaciones SIP usan el puerto 5060 para los protocolos de transporte UDP y TCP, aunque puede utilizar otros protocolos de transporte.
- Uso de URIs (*Uniform Resource Identifier*).
    - Ejemplos de SIP URIs son:
        - sip:usuarioA@<host>
        - sip:usuarioA:<clave>@ing.unlpam.edu.ar
        - sip:usuarioA@ing.unlpam.edu.ar:<puerto>

      Dónde:

        - <host> representa la dirección IP o un nombre de dominio donde se encuentra actualmente registrado el usuarioA.
        - <clave> es un código de seguridad que permite autentificar al usuarioA dentro del dominio.
        - El <puerto> es donde deben enviarse las peticiones y se encuentra asociado el servicio SIP para el dominio del usuarioA.
- Definición de métodos básicos.
- Los mensajes se agrupan en transacciones y llamadas. Generalmente, el cuerpo de los mensajes contiene descripciones de sesiones multimedia.
- Los códigos de respuesta son similares a los de HTTP. Ejemplo: 200 – OK.
- Localización de usuarios y servidores basada en DNS.
- Cabeceras como método de ampliación.
- SIP ofrece todas las potencialidades y las características comunes de la telefonía de Internet como:
    - Llamada o transferencia de medios.
    - Conferencia de llamada.
    - Llamada en espera.

4.3.1. Elementos de la arquitectura SIP

Dentro de la arquitectura SIP existen elementos que interactúan entre sí para llevar a cabo una comunicación entre dos usuarios. Los elementos básicos de una red basada en una señalización

SIP pueden clasificarse en lógicos y físicos, donde un elemento físico (por ejemplo, una computadora) puede implementar uno o varios elementos lógicos. Los elementos lógicos pueden comportarse como servidor, cliente o ambos simultáneamente. Estos elementos SIP son:

- **Agentes de usuario:**
  - Agentes de usuario cliente (UAC).
  - Agentes de usuario servidor (UAS).
  - Agente de usuario back-to-back (B2BUA).
- **Servidores:**
  - De registro o *registrar*.
  - Proxy.
  - De redirección.

Agente de usuario: Son elementos lógicos dentro de la arquitectura SIP. Son los puntos extremos del protocolo, ya que son los que generan y procesan los mensajes del mismo. Los agentes de usuario son definidos por el RFC 3261 (J. Rosenberg, 2002) como una aplicación que contienen dos elementos, un agente de usuario cliente (UAC) y un agente de usuario servidor (UAS). El UAC se encarga de iniciar solicitudes SIP hacia la red y el UAS recibe una solicitud desde la red y se pone en contacto con el usuario para devolverle la respuesta que desea.

Un agente de usuario back-to-back (B2BUA) recibe peticiones y las procesa como un agente de usuario servidor. Para determinar cómo responder a estas peticiones se comporta como un agente de usuario cliente y genera a su vez nuevas peticiones. La diferencia con un servidor proxy es que éste debe mantener el estado del dialogo y participar en todas las peticiones que se realicen en el mismo. Su uso más común es para funciones donde es necesario controlar el saldo del usuario o el tiempo restante que le queda a la comunicación, como es el caso de los locutorios y de los sistemas de línea de control.

Servidor de registro o *registrar*: Se encargan de mantener la información de la ubicación actual del usuario, los cuales deberán registrarse en este servidor. Como ventaja permite una movilidad de los usuarios que podrán ser contactados sin importar donde se encuentren.

Servidor Proxy: Se encarga de encaminar las peticiones y respuestas del protocolo de señalización hasta el destino final. Estas peticiones y respuestas poseen un parámetro denominado *Via* que incluye todos los saltos por donde han viajado, de esta forma se evitan bucles y a su vez se fuerza a que las respuestas viajen por el mismo camino que las peticiones.

Servidores de redirección: Su funcionamiento es similar al servidor proxy. Sin embargo, este no retransmite los mensajes hasta llegar a destino, sino que le indica al origen de la petición la dirección del destino o de otro servidor que puede acercarlo al destino de la comunicación.

Un servidor puede comportarse como proxy o de redirección dependiendo de la situación. Su funcionamiento puede compararse con la resolución de nombres recursiva e iterativa de DNS respectivamente.

4.3.2. Mensajes SIP

SIP es un protocolo basado en texto que utiliza el conjunto de caracteres UTF-8 y sus mensajes son tanto una petición desde un cliente hacia un servidor, o una respuesta de un servidor hacia un cliente. Ambos mensajes utilizan el formato básico definido en el RFC 2822 (Resnick, 2001) con algunas diferencias, SIP permite algunos campos en el encabezado que no serían válidos en la definición del RFC 2822 (Resnick, 2001). Los mensajes consisten de una línea inicial, de uno o más campos de encabezado, una línea vacía indicando el final de los campos del encabezado y un cuerpo de mensaje opcional. La línea inicial puede ser una *línea Request* en caso de una petición o una *línea Status* para una respuesta.

4.3.2.1. Peticiones (Requests)

Las peticiones SIP se distinguen a través de la línea Request en la línea inicial del mensaje. Una línea Request contiene un nombre de método, una dirección URI de petición y una versión de protocolo separados por un solo carácter de espacio.

- **Método:** Se pueden especificar alguno de los siguientes 6 métodos.
  - **Register:** Para registrar información de contacto.
  - **Invite, ACK, Cancel:** Para el establecimiento de sesiones.
  - **Bye:** Para terminar sesiones.

- **Options:** Para consultar a servidores acerca de sus capacidades.

- **Petición URI (Request-URI):** Indica el usuario o servicio al cual se dirige la petición.
- **Versión:** Indica la versión del protocolo SIP que se está utilizando. Incluido tanto en las peticiones como en las respuestas.

4.3.2.2. Respuestas (Responses)

Las respuestas SIP son diferenciadas de las peticiones por medio de la línea Status en su línea inicial. Una línea Status consiste de la versión del protocolo seguido de un código de estado numérico y una descripción textual asociado a dicho código. Cada uno de estos elementos está separado por un carácter de espacio en blanco.

El código de estado es una cifra de tres dígitos enteros que indica la salida de haber intentado comprender y satisfacer una petición. La descripción textual intenta dar una explicación breve del código de estado obtenido. El propósito de este código es que sea usado por sistemas autómatas, mientras que la descripción está dirigida al usuario humano.

El primer dígito del código de estado define la clase de respuesta y los siguientes dos dígitos no tienen ningún rol de categorización. En SIP 2.0 se permiten 6 valores para el primer dígito:

- **1XX:** Provisional. La petición fue recibida y se continúa con el procesamiento de la petición.
- **2XX:** Exitoso. La acción fue recibida, comprendida y aceptada exitosamente.
- **3XX:** Redirección. Se requieren de más acciones para completar la petición.
- **4XX:** Error de cliente. La petición contiene una sintaxis inválida o no puede ser cumplida en este servidor.
- **5XX:** Error de servidor. El servidor falló al cumplir una petición aparentemente válida.
- **6XX:** Fallas globales. La petición no puede ser cumplida en ningún servidor.

4.3.2.3. Campos de encabezado (header fields)

Los campos de encabezado SIP son similares a los de HTTP tanto en sintaxis como en semántica. Cada uno de los campos del encabezado consiste de un nombre de campo seguido de dos puntos ":" y un valor del campo. Además, a los campos de encabezado se les puede asociar parámetros

que aparecen luego del valor del campo, separados por un punto y coma ";". Los parámetros están formados por un nombre de parámetro y un valor para dicho parámetro separados por el carácter "=".

4.3.2.4. Cuerpo de mensaje

Las peticiones podrían tener un cuerpo de mensaje y su interpretación dependerá del método indicado en la petición. En el caso de las respuestas, será interpretado según el método de petición y el código de estado. El tipo de cuerpo de mensaje debe ser indicado por el campo de encabezado Content-Type, el cuál indicará además, si es posible, el conjunto de caracteres utilizado en él. En caso de que el cuerpo de mensaje sea codificado, por ejemplo para compresión, el Content-Type debe ser ignorado y el campo Content-Encoding debe indicar esta situación.

4.3.3. Métodos del protocolo SIP

Los métodos utilizados dentro del protocolo SIP son definidos en el RFC 3261 (J. Rosenberg, 2002). Estos métodos son las operaciones principales que realizan los distintos componentes descriptos en el RFC del protocolo para llevar a cabo la señalización de las sesiones multimedia. Posteriormente, y para la adaptación del protocolo con nuevas tecnologías, los métodos fueron adaptados y extendidos en nuevos RFCs donde se describen los nuevos métodos del protocolo. Los métodos principales son:

- Register
- Invite
- Re-invite
- ACK
- Bye
- Cancel
- Options

Método Register: El método de registración consiste en enviar una petición Register a un tipo especial de UAS conocido como servidor de registración o *Registrar*. Este tipo de servidor, como se mencionó anteriormente, actúa como servicio de localización de usuarios para los dominios, leyendo y escribiendo los mapeos basados en el contenido de las peticiones Register. Este tipo de

servicio es utilizado generalmente por los servidores proxy para los pedidos de enrutamiento durante la inicialización de sesiones dentro o fuera de un dominio.

Método Invite: Cuando un usuario cliente desea iniciar una sesión multimedia con otro (para audio, vídeo, o un juego), envía una petición Invite. Esta petición pide establecer una sesión a un servidor y puede ser re-enviada por los servidores proxy que se encuentren en el camino entre los extremos de la sesión, llegando eventualmente a unos o más UAS que puedan potencialmente aceptar o no la invitación. Estos UASs necesitarán con frecuencia consultar al usuario si acepta la invitación. En caso afirmativo, la sesión debe ser establecida y se envía una respuesta 2xx. Si la invitación no es acepta por el usuario, se envía una respuesta 3xx, un 4xx, un 5xx o 6xx, dependiendo de la razón del rechazo. Antes de enviar una respuesta final, el UAS puede también enviar respuestas provisorias (1xx) para indicar al UAC del progreso de contacto con al usuario llamado.

Método Re-invite: Mediante este método se puede indicar el cambio de direcciones o puertos, agregar un flujo multimedia, suprimir otro, etc. Esto se logra enviando una nueva petición Invite dentro de la misma sesión que fue establecida anteriormente. Una petición Invite enviada dentro de la misma sesión se conoce como re-Invite.

Para modificar la sesión y los parámetros de la misma se necesita solamente de un mensaje Re-invite. Pueden generarse problemas si no se conoce exactamente qué está sucediendo entre los UAC y UAS de la sesión en el momento en que se envía este tipo de mensaje.

Método ACK: Mediante este método se indica la aceptación dentro de la sesión para iniciar un canal de comunicación. SIP implementa una comunicación en tres pasos, como lo hace el protocolo TCP:

1. El usuario que requiere iniciar una llamada envía un Invite.
2. El usuario que recibe la llamada envía un mensaje 200 - OK para aceptar la llamada.
3. El que inicia la llamada envía un ACK para confirmar el 200 - OK y establecer la llamada, terminando así el proceso.

Método Cancel: La petición Cancel, se utiliza para cancelar una petición anteriormente enviada por un cliente. Específicamente, pide al UAS que deje de procesar la petición y que genere una respuesta de error para la misma. El método Cancel no tiene ningún efecto sobre peticiones a las cuales un UAS ya ha dado una respuesta final.

El método Cancel es lo mejor para las peticiones Invite, que pueden tomar un tiempo largo para generar una respuesta. En ese caso, un UAS que recibe un pedido de Cancel a un Invite que todavía no ha enviado una respuesta final, deja de esperar la respuesta del usuario que recibe la llamada y responde el Invite con una respuesta de error específica (487).

Método Bye: Este método es utilizado para terminar una sesión específica. Cuando un Bye se recibe en una comunicación, cualquier sesión asociada a ésta debe terminar.

Método Options: Todos los UAS deben soportar el método Options. Este método permite a un agente de usuario preguntar a otro o a un servidor proxy sobre sus capacidades. Esto permite que un cliente descubra información sobre los métodos soportados, los tipos de contenidos, las extensiones, los codecs, etc. sin tener que provocar el "ringing" de la otra parte. Por ejemplo, antes de que un cliente inserte en el método Invite una opción que no es seguro que sea soportada por el UAS destino en el encabezado Require, el cliente puede preguntarle a éste a través de un mensaje Options para ver si esta opción es devuelta en la respuesta en el campo del encabezado Supported.

4.3.4. Comportamiento de los agentes de usuario

Un agente de usuario representa un usuario final, el cual contiene un agente de usuario cliente, encargado de generar peticiones, y un agente de usuario servidor, encargado de responderlas. Un UAC es capaz de generar una petición basado en un estímulo externo y procesar una respuesta. Un UAS es capaz de recibir una petición y de generar una respuesta basada en una entrada de usuario, estímulos externos, el resultado de la ejecución de algún programa o en algún otro mecanismo.

Cuando un UAC envía una petición, ésta viaja a través de un número de servidores proxy los cuales reenvían la petición hasta un UAS. Cuando el UAS genera la respuesta, ésta es dirigida hacia el UAC

por el mismo camino que la petición ha registrado en el campo Via.

Los procesos del UAC y UAS dependen fuertemente de dos factores. Primero, si la petición o la respuesta se encuentran dentro o fuera de un diálogo (una relación SIP extremo-a-extremo entre dos agentes de usuario que persiste durante un tiempo), y segundo, del método de la petición (Register, Invite, Cancel, etc.).

4.3.4.1. Proceso de registración y ubicación de usuarios

SIP posee la capacidad de descubrimiento de usuarios conocida como servicio de localización. Cuando un usuario desea iniciar una sesión con otro, primero se debe conocer la ubicación actual del destino de la comunicación para que ésta pueda llevarse a cabo. Este proceso se logra a través de servidores proxy y de redirección, que son los encargados de recibir una petición, determinar dónde debe ser enviada y finalmente enviarla. Para esto, estos servidores deben proveer direcciones vinculadas a un dominio en particular, las cuales deben ser relacionadas con las SIP o SIP URIs solicitadas (esquemas particulares del protocolo SIP para identificación de recursos universales) con una o más URIs “cercanas” al usuario deseado. Los servidores proxy consultan al servicio de localización, que relaciona la URI recibida con el agente de usuario, dónde se encuentra el destinatario deseado.

La registración es la operación por la cual se crean las vinculaciones del servicio de localización para un dominio en particular, que asocia un registro de dirección URI con una o más direcciones de contacto. De esta forma, cuando un proxy recibe para ese dominio una petición cuyo Request-URI coincide con algún registro de dirección, enviará la petición a la dirección de contacto asociada con ese registro.

La operación de registración implica enviar una petición Register a un servidor de registro. El servidor de registro actúa como una interfaz de usuario para el servicio de localización para el dominio, leyendo y escribiendo relaciones basadas en el contenido de la petición Register. Este servicio de localización es consultado luego por el servidor proxy que es el responsable de encaminar los requerimientos para ese dominio.

Para ilustrar esta situación, se dará nombre a los siguientes elementos con el fin de ejemplificar el proceso de registración y ubicación de usuarios a través del protocolo SIP:

- Usuario A: Usuario que será localizado para iniciar una llamada.
- Usuario B: Usuario que iniciará una llamada.
- ingenieria.edu.ar: Dominio al que pertenecen los usuarios A y B.
- rectorado.ingenieria.edu.ar: Host en el que se encuentra el usuario A.
- difusion.ingenieria.edu.ar: Host en el que se encuentra el usuario B.
- sip1.ingenieria.edu.ar: Servidor proxy dentro del dominio ingenieria.edu.ar.
- sip2.ingenieria.edu.ar: Servidor de registración dentro del dominio ingenieria.edu.ar.
- sip3.ingenieria.edu.ar: Segundo servidor proxy dentro del dominio ingenieria.edu.ar.

El proceso de registración y ubicación, representado por la Figura 4.3.1, será el siguiente:

1. El UAC del usuario A registra su ubicación en el servidor sip2.ingenieria.edu.ar a través del método Register.
2. El servidor de registración sip2.ingenieria.edu.ar crea un enlace en el servicio de localización entre la ubicación del usuario A y la dirección de contacto para el mismo (sip:usuarioA@ingenieria.edu.ar).
3. El UAC del usuario B inicia una llamada hacia el usuario A mediante el método Invite enviando una petición al servidor proxy sip1.ingenieria.edu.ar.
4. El servidor proxy sip1.ingenieria.edu.ar consulta al servidor más cercano, en este caso el servidor proxy sip3.unlpam.edu.ar, por la dirección de contacto <usuarioA@ingenieria.edu.ar>.
5. El servidor proxy sip3.ingenieria.edu.ar genera una consulta al servicio de localización para la dirección de contacto usuarioA@ingenieria.edu.ar.
6. El servicio de localización procesa la consulta y retorna la ubicación actual del usuario A, es decir, usuarioA@rectorado.ingenieria.edu.ar.
7. El servidor proxy sip3.ingenieria.edu.ar propaga la petición Invite a dicha ubicación.

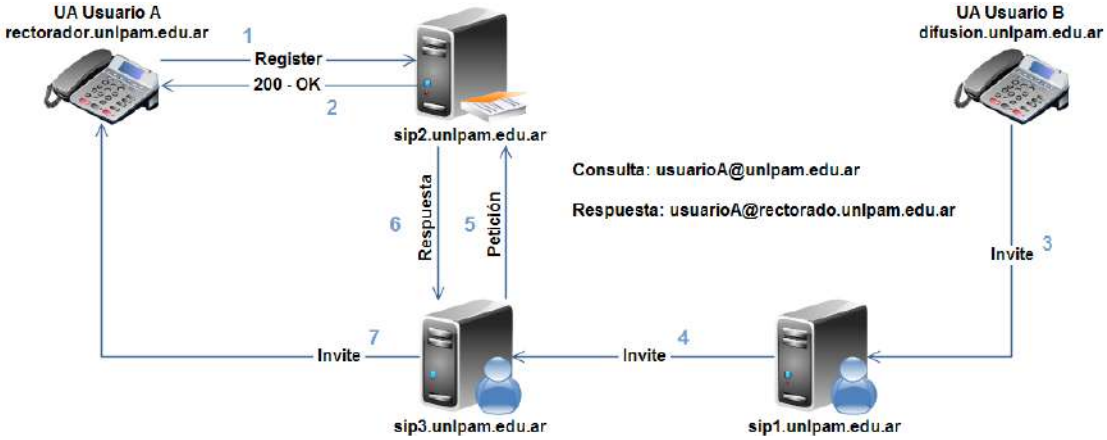


Figura 4.3.1: Registración y ubicación de usuarios SIP.

Los mensajes SIP de petición y respuesta para la registración de usuarios son:

1. Petición Register del UA del usuario A al servidor proxy sip2.ingenieria.edu.ar:

   Register sip:sip2.ingenieria.edu.ar SIP/2.0

   Via: SIP/2.0/UDP rectorado.ingenieria.edu.ar:5060 ;branch=z9hG4bKnashds7

   Max-Forwards: 70

   To: Usuario A <sip:usuarioA@ingenieria.edu.ar>

   From: Usuario A<sip:usuarioA@ingenieria.edu.ar>;tag=456248

   Call-ID: 843817637684230@998sdasdh09

   CSeq:1826REGISTER

   Contact:<sip:usuarioA@192.0.2.4>

   Expires: 7200

   Content-Length: 0

La petición *URI* inicial (*Request-URI*) del mensaje nombra el dominio del servicio de localización donde la registración será realizada.

El campo ***Via*** indica que utilizará el protocolo UDP para transportar los datos y que rectorado.ingenieria.edu.ar es donde debe ser enviada la respuesta de la petición. El parámetro *branch* es utilizado para identificar la transacción creada por la petición. Este parámetro es obligatorio y siempre debe comenzar con los caracteres “z9hG4bK”.

Max-Forwards es la cantidad de saltos máxima que la petición puede atravesar, disminuyendo una unidad su valor en cada salto.

To contiene la dirección de registro de quién está siendo registrado.

From contiene la dirección de registro de la persona responsable para la registración. El valor es el mismo del campo *To* a menos que la petición se realice por un tercero.

Call-ID actúa como identificador único y debe ser el mismo para todas las peticiones y respuestas enviadas por los UAs en el díalogo.

CSeq sirve para identificar y ordenar las transacciones. Consiste de un número de secuencia y un método, el cual coincide con el método de la petición.

Contact contiene las direcciones enlazadas. Puede contener ninguno o varios valores.

Expires indica el tiempo válido en segundos que el UA puede enlazar la dirección de registro con la dirección de contacto.

Content-Length indica el tamaño del mensaje en bytes.

2. Respuesta de estado de la sesión del UAS al servidor proxy sip3.ingenieria.edu.ar:

   SIP/2.0 200 OK

   Via: SIP/2.0/UDP rectorado.ingenieria.edu.ar:5060

   ;branch=z9hG4bKnashds7;received=192.0.2.4

   To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=2493k59kd

   From: Usuario A<sip:usuarioA@ingenieria.edu.ar>;tag=456248

   Call-ID: 843817637684230@998sdasdh09

   CSeq:1826REGISTER

   Contact:<sip:usuarioA@192.0.2.4>

   Expires: 7200

   Content-Length: 0

El ejemplo anterior ilustra la ubicación de un usuario dentro de un mismo dominio SIP para luego comenzar con el proceso de comunicación. Si el usuario destino de la comunicación se encuentra

fuera del dominio local, lo primero a resolver es donde deben enviarse las peticiones SIP en el dominio destino. La ubicación del servidor SIP se realiza a través de una petición DNS generalmente a un registro SRV. Un servidor SIP puede administrar más de un dominio SIP.

Para que el servidor DNS del dominio destino pueda responder a estas peticiones debe existir una entrada en su archivo de registros similar a la siguiente:

```
_sip._udp.<DOMINIO>. <TTL> IN SRV <PRIORIDAD><PESO><PUERTO><SERVIDOR>
```

Para el tipo de registro SRV lo primero a establecer es el nombre simbólico del servicio y el protocolo de transporte que utilizará dicho servicio, con_sip._udp. se establece que el registro SRV corresponde al servicio SIP sobre el protocolo de transporte UDP. Si en lugar de utilizar el protocolo de transporte UDP se requiere TCP, debería comenzar de la forma _sip._tcp.

Otra forma en la que pueden ubicarse usuarios es a través de ENUM (**Telephone number mapping**). ENUM es el proceso por el cual a un número telefónico se lo enlaza a una dirección Internet la cual está registrada en un sistema DNS utilizando un tipo de registro DNS especial denominado NAPTR, en una zona específica denominada e164.arpa. Por lo tanto, un número ENUM se registra de la misma forma como si se tratase de un dominio en Internet, y por ende cuando otro usuario quiere comunicarse con otro que posee un número ENUM, la resolución del destino se obtiene a través de peticiones sobre DNS. Estos números ENUM están organizados bajo el estándar E.164.

4.3.4.2. Inicio de sesión

Una vez conocidas las ubicaciones de los usuarios que participarán en una llamada, la petición Invite es enviada al destino con el fin de establecer una sesión. Siguiendo con el ejemplo anterior, se explicará el proceso de inicio de sesión el cual será reflejado en la Figura 4.3.2:

1. El UAC del usuario B envía un Invite al servidor proxy sip1.ingenieria.edu.ar.

2. El servidor proxy sip1.ingenieria.edu.ar retrasmite el Invite al servidor proxy sip3.ingenieria.edu.ar y retorna un mensaje de estado de sesión 100 – Trying al usuario B indicando que se está intentando realizar la comunicación.

3. El servidor proxy sip3.ingenieria.edu.ar recibe el Invite y retransmite la petición hacia la última ubicación del usuario A registrada por el método Register del ejemplo anterior. Para conocer dicha ubicación, deberá consultar al servicio de localización. Además, envía al servidor proxy sip1.ingenieria.edu.ar un mensaje de estado de sesión 100 – Trying indicando que se está intentando realizar la comunicación.

4. El UAS del usuario A recibe la petición de Invite y emite una respuesta 180 – Ringing indicándole al servidor proxy sip3.ingenieria.edu.ar que la petición llegó a destino.

5. El servidor proxy sip3.ingenieria.edu.ar retransmite la respuesta 180 – Ringing al servidor proxy sip1.ingenieria.edu.ar.

6. El servidor proxy sip1.ingenieria.edu.ar retransmite la respuesta 180 – Ringing al UAC del usuario B.

7. Cuando el usuario A acepta la llamada, el UAS emite la respuesta 200 – Ok para indicar que la petición fue aceptada con éxito.

8. El servidor proxy sip3.ingenieria.edu.ar retransmite la respuesta 200 – Ok al servidor proxy sip1.ingenieria.edu.ar.

9. El servidor proxy sip1.ingenieria.edu.ar retransmite la respuesta 200 – Ok al UAC del usuario B.

10. El UAC del usuario B emite una respuesta ACK para indicar que recibió correctamente la contestación al UAS del usuario A. Este mensaje se envía directamente entre los agentes de usuario sin la intervención de los servidores proxies. Esto se debe a que en este punto los extremos finales ya conocen la dirección del otro por medio de los mensajes intercambiados.

11. La sesión multimedia queda establecida y se transmite extremo-a-extremo sin la intervención de los servidores proxies.

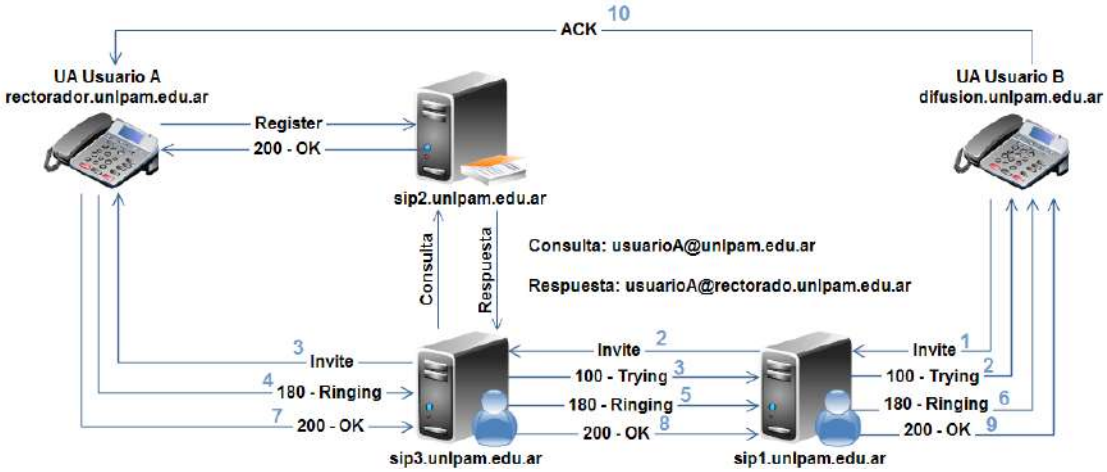


Figura 4.3.2: Inicio de sesión SIP.

Los mensajes SIP de petición y respuesta intervinientes en el establecimiento de la sesión SIP del ejemplo anterior tienen el siguiente aspecto:

1. Petición Invite del UAC del usuario B al proxy sip1.ingenieria.edu.ar:

INVITE sip:usuarioA@ingenieria.edu.ar SIP/2.0

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar ;branch=z9hG4bKnashds8

Max-Forwards: 70

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>

From: Usuario B <sip:usuarioB@ingenieria.edu.ar> ;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioB@difusion.ingenieria.edu.ar>

Content-Type: aplication/sdp

Content-Length: 142

La petición *URI* inicial del mensaje es establecida con el valor URI indicado por el campo ***To***.

To especifíca el destinatario lógico deseado de la petición, o el registro de dirección del usuario o recurso al que se requiere contactar.

From indica la entidad lógica del origen de la petición. "Usuario B" es el nombre que se mostrará en el UAS del usuario A y es opcional. El parámetro *tag* identifica a este UA como un extremo del diálogo.

Contact provee un SIP o SIPS URI que puede ser usado por el UA del usuario A para contactar al UA del usuario B para las siguientes peticiones.

Content-Type indica el tipo de medio del cuerpo del mensaje enviado al destinatario. En este caso SDP.

2. Respuesta de estado de la sesión del servidor proxy sip1.ingenieria.edu.ar para la petición anterior:

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP difusion.ingenieria.edu.ar
   ;branch=z9hG4bKnashds8;receiver=192.0.2.1
   To: Usuario A <sip:usuarioA@ingenieria.edu.ar>
   From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774
   Call-ID: a84b4c76e66710
   CSeq: 314159 INVITE
   Content-Length: 0

3. Petición Invite del proxy sip1.ingenieria.edu.ar al servidor proxy sip3.ingenieria.edu.ar:

   INVITE sip:usuarioA@ingenieria.edu.ar SIP/2.0
   Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;branch=z9hG4bK77ef4c2312983.1
   Via: SIP/2.0/UDP difusion.ingenieria.edu.ar
   ;branch=z9hG4bKnashds8;receiver=192.0.2.1

Max-Forwards:69

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact: <sip:usuarioB@difusion.ingenieria.edu.ar>

Content-Type: aplication/sdp

Content-Length: 142

4. Respuesta de estado de la sesión del servidor proxy sip3.ingenieria.edu.ar para la petición anterior:

SIP/2.0 100 Trying

Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;

branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Content-Length: 0

5. Petición Invite del proxy sip3.ingenieria.edu.ar al UA del usuario A:

INVITE sip:usuarioA@192.0.2.4 SIP/2.0

Via: SIP/2.0/UDP sip3.ingenieria.edu.ar;branch=z9hG4bK4b43c2ff8.1

Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;

branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

Max-Forwards:68

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>

From: Usuario B <sip:usuarioB@ingenieria.edu.ar> ;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioB@difusion.ingenieria.edu.ar>

Content-Type: aplication/sdp

Content-Length: 142

6. Respuesta de estado de la sesión del AUS al servidor proxy sip3.ingenieria.edu.ar para la petición anterior:

SIP/2.0 180 Ringing

Via: SIP/2.0/UDP sip3.ingenieria.edu.ar;

branch=z9hG4bK4b43c2ff8.1;receiver=192.0.2.3

Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;

branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioA@192.0.2.4>

Content-Length: 0

7. Respuesta de estado de la sesión del servidor proxy sip3.ingenieria.edu.ar al servidor proxy sip1.ingenieria.edu.ar:

SIP/2.0 180 Ringing

Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;

branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioA@192.0.2.4>

Content-Length: 0

8. Respuesta de estado de la sesión del servidor proxy sip1.ingenieria.edu.ar al UAC del usuario B:

SIP/2.0 180 Ringing

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioA@192.0.2.4>

Content-Length: 0

9. Respuesta de estado de la sesión del AUS al servidor proxy sip3.ingenieria.edu.ar:

SIP/2.0 200 OK

Via: SIP/2.0/UDP sip3.ingenieria.edu.ar;

branch=z9hG4bK4b43c2ff8.1;receiver=192.0.2.3

Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;
branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2
Via: SIP/2.0/UDP difusion.ingenieria.edu.ar
;branch=z9hG4bKnashds8;receiver=192.0.2.1
To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf
From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact:<sip:usuarioA@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

10. Respuesta de estado de la sesión del servidor proxy sip3.ingenieria.edu.ar al servidor proxy sip1.ingenieria.edu.ar:

SIP/2.0 200 OK
Via: SIP/2.0/UDP sip1.ingenieria.edu.ar;
branch=z9hG4bK77ef4c2312983.1;receiver=192.0.2.2
Via: SIP/2.0/UDP difusion.ingenieria.edu.ar
;branch=z9hG4bKnashds8;receiver=192.0.2.1
To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf
From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact:<sip:usuarioA@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

11. Respuesta de estado de la sesión del servidor proxy sip1.ingenieria.edu.ar al UAC del usuario B:

SIP/2.0 200 OK

Via: SIP/2.0/UDP difusion.ingenieria.edu.ar

;branch=z9hG4bKnashds8;receiver=192.0.2.1

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 INVITE

Contact:<sip:usuarioA@192.0.2.4>

Content-Type: application/sdp

Content-Length: 131

12. Respuesta del UAS del usuario B al usuario A:

ACK sip:usuarioA@192.0.2.4 SIP/2.0

Via: SIP/2.0/UDP rectorado.ingenieria.edu.ar ;branch=z9hG4bKnashds9

Max-Forwards:70

To: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=a6c85cf

From: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 314159 ACK

Content-Length: 0

4.3.4.3. Finalización de la sesión

Para finalizar la sesión multimedia, los extremos de la comunicación deberán terminar el dialogo del protocolo de señalización. Siguiendo con el ejemplo anterior, el intercambio de mensajes para tal finalidad se muestra en la figura xxxxxx que contiene los siguientes pasos:

1. El agente de usuario utiliza el método Bye para finalizar la comunicación y envía una petición al UAS del usuario B.
2. El UAC del usuario B responde a la petición con un mensaje de estado 200 – OK y cierra los canales de comunicación.

Los mensajes SIP de petición y respuesta intervinientes en la finalización de la sesión SIP tienen el siguiente aspecto:

1. El UAS del usuario A envía la Petición Bye al UAC del usuario B:

BYE sip:usuarioB@difusion.ingenieria.edu.ar SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
Max-Forwards: 70
To: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=a6c85cf
From: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq:231BYE
Content-Length: 0

2. Respuesta del UAC del usuario B al UAS del usuario A:

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
To: Usuario B <sip:usuarioB@ingenieria.edu.ar>;tag=a6c85cf
From: Usuario A <sip:usuarioA@ingenieria.edu.ar>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq:231BYE
Content-Length: 0

4.3.5. Desventajas del protocolo SIP

SIP es un protocolo de señalización estandarizado por la IETF lo cual lo hace ampliamente implementado por la mayoría los fabricantes de equipos y software. Sin embargo, al ser un protocolo de propósito general para poder transmitir sin dificultad cualquier información y no sólo audio o video, se producen algunos inconvenientes cuando se lo quiere adaptar a un entorno VoIP complejo.

Uno de estos problemas surge debido a que SIP es utilizado únicamente para la señalización y es necesario utilizar un protocolo adicional para el transporte de los datos multimedia. Los datos de la conexión requeridos por este protocolo adicional serán enviados a través de SIP (dirección IP y puerto). En ocasiones, puede suceder que alguna de las partes intervinientes en la comunicación se encuentre detrás un Firewall o NAT que bloqueé o traduzca el puerto anunciado por SIP, donde funcionará el protocolo adicional para el flujo de los datos multimedia, a otro puerto distinto impidiendo concretar la conexión de manera satisfactoria. Esto es una importante desventaja del protocolo SIP que impide el intercambio de información multimedia en determinados ambientes de red y obliga a tener que recurrir a mecanismos adicionales para poder proveer una solución que hace más compleja y costosa la comunicación. Además, al utilizar dos protocolos por separado se involucran más puertos en la comunicación. Para señalización se utilizará el puerto 5060 y dos puertos RTP para cada conexión de audio. Por ejemplo, para 100 llamadas simultáneas con SIP se usarían 200 puertos RTP más el puerto de señalización.

Las pérdidas de mensajes SIP pueden ser detectadas luego de un tiempo considerado inapropiado para el rendimiento esperado en ambientes VoIP. Debido que SIP trabaja bajo el protocolo de transporte UDP, y al ser éste un protocolo que no está orientado a la conexión, no cuenta con un mecanismo inherente para la recuperación de información ante la pérdida de la misma.

Otra desventaja es que SIP no respeta la separación de capas propuesta por el modelo OSI. Esto se debe a que este protocolo transporta la dirección IP de las partes involucradas en la comunicación dentro de los mensajes de señalización. Ante la inminente migración de la versión actual del protocolo de red IP (IPv4) a la versión 6 (IPv6), SIP necesitará adaptarse al nuevo entorno.

Para solucionar estos problemas, la IETF introdujo nuevos protocolos que trabajan en conjunto con SIP. Sin embargo, estas inclusiones le generan complejidad adicional a los diseñadores e implementadores del estándar. Más aun, surgen problemas de interoperabilidad y compatibilidad entre las redes de proveedores de servicios que resultan críticos. Por esta razón, es necesario explorar nuevas alternativas que permitan un mejor desempeño en la resolución de estos problemas.

4.4. Protocolo IAX/IAX2

Las iniciales de IAX están dadas por **I**nter-**A**steriske**X**change. Este protocolo fue desarrollado en el contexto del proyecto Asterisk (una central telefónica de código abierto para VoIP), aunque no es específico de ésta plataforma. Sin embargo, puede ser utilizado independientemente de Asterisk para controlar la señalización y los datos multimedia entre dos extremos (UAS/UAC). La definición de este protocolo se encuentra en el RFC 5456 (M. Spencer, 2010).

IAX es un protocolo "acoplado" al camino que es utilizado tanto para señalización como operaciones de control multimedia (audio, video y datos). Provee características interesantes tales como administración de señalización y transferencia multimedia, soporte para funciones de aprovisionamiento nativo y mantenimiento de firmware. Tiene la ventaja de solucionar los problemas que se presentan con NAT en SIP, ya que es independiente del protocolo IP.

Las características principales de IAX son:

- Funciona sobre el protocolo UDP utilizando un único puerto (por defecto el 4569).
- Es un protocolo binario, lo que permite optimizar el ancho de banda utilizado mejorando la calidad de las llamadas.
- Una llamada IAX puede consistir de varios segmentos. Cada uno de estos puede ser implementado utilizando diferentes protocolos. Por ejemplo, de SIP a IAX y de IAX a ISDN.
- Es un protocolo peer-to-peer (P2P) optimizado. Si dos segmentos adyacentes utilizan el protocolo IAX y el nodo intermediario determina que no es necesario permanecer en el camino de la llamada, puede removerse del mismo.
- Incluye un mecanismo de registración donde el registrante debe contactarse con un servidor de registro.
- No requiere protocolos adicionales para intercambiar flujos multimedia, ya que lo maneja por sí mismo. Dentro de los tipos multimedia que puede enviar se encuentran: audio, video, texto, imágenes, HTML, etc.
- Define dos tipos de mensajes: confiables y no confiables. Los mensajes no confiables son aquellos flujos los cuales no deben confirmarse su recepción ni deben ser retransmitidos

en caso de perderse. Los mensajes confiables deberían ser confirmados al ser recibidos, de lo contrario serán retransmitidos.

- Ninguna dirección IP es encapsulada en los mensajes de señalización en el protocolo, brindando una solución a NAT.

- Define un conjunto de mensajes que son utilizados para monitorear el estado de la red, que pueden ser enviados durante una llamada activa como también por fuera de la misma.

- Ofrece los medios necesarios para verificar si un participante de una llamada remota está activo o no.

- Permite implementar métodos de seguridad IP nativos como así también el intercambio de claves compartidas. Esto puede utilizarse tanto con texto plano como en conjunto de mecanismos de encriptación como AES.

A pesar de que este protocolo es muy efectivo satisfaciendo muchas de las necesidades de comunicación actuales, sufre de algunas limitaciones. Por ejemplo, dado que IAX utiliza un mecanismo de negociación de codecs punto-a-punto, tiene problemas de extensibilidad ya que cada nodo interviniente en el camino de la llamada debe soportar en cierto grado cada códec utilizado. Además, la definición del códec es controlada por una máscara interna de 32 bits, por lo tanto, estos deben ser definidos en el protocolo y está limitado el número máximo de codecs que pueden ser utilizados en simultáneo.

Por otra parte, una de las principales ventajas de diseño de IAX es además un potencial problema. El uso de un único puerto conocido (4569) hace que este protocolo sea un blanco fácil para ataques de denegación de servicio. Particularmente, los sistemas de VoIP son sensibles a este tipo de ataques. Este protocolo ha sido diseñado de tal forma que todo el tráfico multimedia y de señalización es dirigido a un servidor centralizado. Mientras que este enfoque provee una gran ventaja para el control, genera a la vez un problema de escalabilidad en todo el sistema. Actualmente IAX provee también la posibilidad de separar el flujo multimedia del de señalización, lo que de alguna forma ayuda a superar esta limitación.

La mayoría de las desventajas que sufre IAX son debido a problemas en su implementación más que al protocolo en sí. Esto suele deberse, por ejemplo, a la limitación de hilos habilitados para procesar el tráfico IAX, lo que puede llevar a ataques denegación de servicios cuando se supera éste número.

4.4.1. Identificadores de Recursos Universales IAX (IAX URI)

Para realizar una llamada, el protocolo IAX utiliza un esquema URI propio el cual identifica un recurso de comunicación. Las URIs contienen suficiente información para poder iniciar una llamada IAX con dicho recurso. Por ejemplo, una URI IAX puede representar una casilla de voice mail en un servicio de mensajería, un programa interactivo, una dirección de red PSTN, etc.

El esquema URI IAX representa una ubicación que podría ser utilizada por el protocolo IAX para establecer una nueva llamada utilizando los componentes del mismo.

La sintaxis del esquema URI IAX está definida siguiendo los lineamientos del estándar RFC 3986 (T. Berners-Lee, 2005). En forma general, se describe de la siguiente manera:

```
iax:[username@]host[:port][/number[?context]]
```

- iax: Nombre del esquema URI.
- username: Cadena de caracteres utilizado para propósitos de identificación.
- host: El dominio del recurso. Puede ser tanto un nombre de dominio completamente calificado (FQDN) o una dirección IPv4 ó IPv6. En caso de tratarse de una dirección IPv6, ésta debe ser encerrada entre corchetes, por ejemplo [2001:db8::1], tal como se define en el estándar antes mencionado. Esta notación es utilizada para poder diferenciar la dirección IP del puerto.
- port: Número de puerto UDP.
- number: El nombre que identifica al recurso en dicho host. Se llama number porque históricamente se trataba de un número, en telefonía IP puede contener tanto números como letras.
- context: Nombre de la partición del host en el cual el servicio es identificado o procesado.

Ejemplos:

- iax: unlpam.edu.ar/alicia
- iax: unlpam.edu.ar:4569/alicia
- iax: unlpam.edu.ar:4569/alicia?oficina
- iax: 200.51.93.227:4569/alicia?oficina
- iax: [2001:db8::1]:4569/alicia?oficina
- iax: unlpam.edu.ar/54554455

4.4.2. Comportamiento entre pares y mensajes relacionados

Todos los mensajes IAX son transportados en Frames, la unidad de comunicación atómica entre dos pares. Pueden distinguirse tres tipos de Frames: “Full Frames”, “Mini Frames” y “Meta Frames”. Los Full Frames transportan información de señalización y control, los Mini Frames solo transportan los flujos multimedia y los Meta Frames son utilizados para armar el camino troncal de la llamada o transmitir flujos de video.

Los mensajes están divididos en dos categorías: Confiables y no Confiables. Los mensajes confiables son conocidos también como Full Frames. Además del indicador del tipo y las facilidades para asegurar la confiabilidad, los mensajes incluyen el identificador de llamada completo, el cual consiste de los identificadores de cada uno de los pares para esa llamada. Además, pueden estar asociados a este tipo de Frame atributos como los elementos de información (IEs), los cuales son utilizados para brindar información adicional. Los mensajes no confiables son conocidos también como Mini Frames y Meta Frames. Estos mensajes solo tienen el identificador del par que inicia la llamada y no debe incluir ningún elemento de información.

Los mensajes que intercambian los pares IAX involucrados en una llamada pueden ser clasificados en las siguientes categorías funciones:

- Registración (opcional).
- Administración de enlace de llamada.
- Optimización del camino de la llamada (opcional).
- Operaciones en el enlace durante la llamada (Mid-Call Link Operations).
- Abandono de llamada (Call Tear Down).
- Monitoreo de red.
- Marcado digital (opcional).

- Varios.
- Mensajes multimedia.

4.4.2.1. Registración

Para que un par IAX pueda comunicarse, el nodo que inicia la llamada debe conocer la dirección de red del destinatario. Esta dirección de red puede ser provista manualmente, determinada a través de un directorio compartido, o ser configurada a través del protocolo IAX. IAX provee un mecanismo para que un nodo pueda registrar su dirección y credenciales con otros nodos para que de esta manera los emisores de las llamadas puedan localizarlo (Figura 4.4.1). Este mecanismo de registración es opcional, y si se implementa puede ser realizado en partes. Además, el protocolo permite la autenticación de usuarios a través de varios métodos, tales como MD5, claves RSA, 3DES, etc.

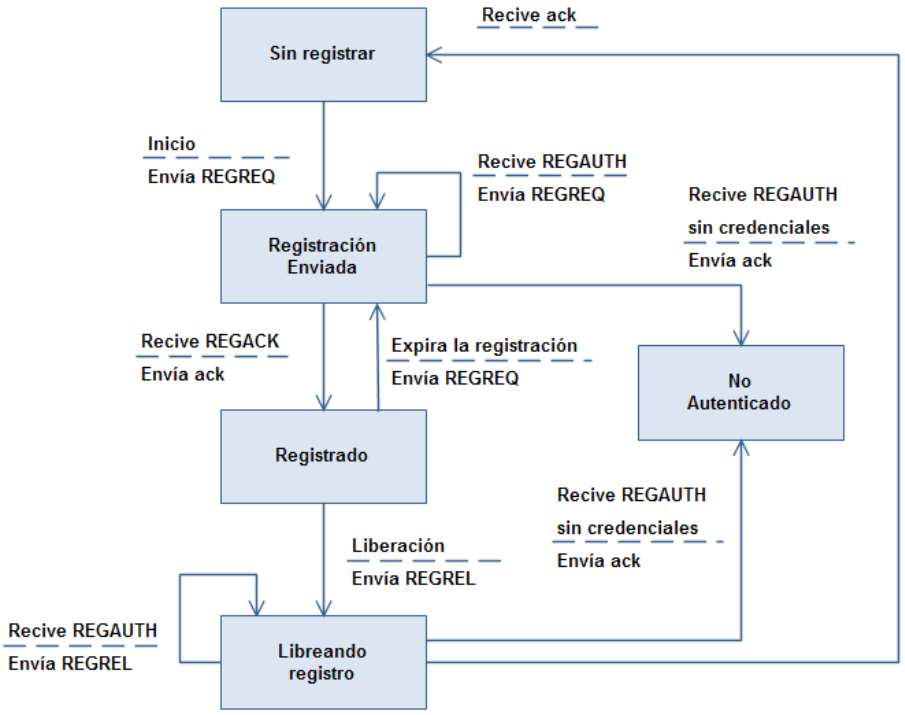


Figura 4.4.1: Registración IAX.

La registración se realiza cuando un registrante envía un nombre de usuario y un pedido de actualización de registración al servidor de registro. Esto se logra con un mensaje REGREQ. Si se requiere autenticación, el servidor registrar responde con un mensaje REGAUTH que indica los tipos de autenticación que soporta. En respuesta el registrante reenvía un REGREQ junto con uno

de los métodos de autenticación soportados. Si el registrante no puede autenticarse, no es necesario llevar a cabo otra acción. Si es aceptada, el servidor registrar envía un mensaje REGACK, el cual debe indicar la "dirección aparente" que identifica al registrante y debería también indicar el "tiempo de expiración" del registro. Si el tiempo de expiración no fue enviado, por defecto los pares deben asumir 60 segundos de expiración. Durante este proceso el servidor registrar puede enviar en cualquier momento un mensaje REGREJ para indicar una falla (Figura 4.4.2).

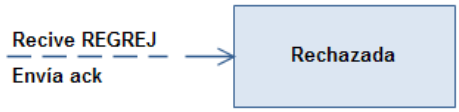


Figura 4.4.2: Registración rechazada.

La registración tiene asociado a ella un periodo de tiempo específico que indica por cuanto tiempo ésta es válida. Este periodo de tiempo comienza cuando el servidor registrar envía el mensaje REGACK. Un registrante puede extender este periodo repitiendo el proceso de registración. Además, puede forzar la expiración en el servidor registrar mediante el envío del mensaje REGREL. Este mensaje puede ser desafiado con un mensaje REGAUTH o, si fueron incluidas suficientes credenciales, ser aceptado con REGACK. En respuesta a un REGAUTH, el registrante debe reenviar el mensaje REGREL utilizando las credenciales especificadas.

4.4.2.2. Administración de enlace de llamada

El protocolo IAX puede ser utilizado para establecer los enlaces o segmentos de la llamada entre dos pares con el propósito de llevar a cabo una llamada. El proceso comienza cuando uno de los pares envía un mensaje NEW indicando el número del dispositivo destino en el par remoto. Este último, puede responder tanto con un desafío de credenciales (AUTHREQ), un mensaje REJECT, o un mensaje ACCEPT. El mensaje AUTHREQ indica el esquema de autentificación permitido y debería resultar en el envío de un mensaje AUTHREP con las credenciales solicitadas. El mensaje REJECT indica que la llamada no puede ser establecida en ese momento. ACCEPT indica que el segmento de llamada entre estos dos pares es establecido y que la señalización de la llamada de más alto nivel puede proceder. Después de enviar o recibir el mensaje ACCEPT, el segmento de llamada queda en estado "enlazado" y es utilizado para pasar mensajes de control de la llamada hasta que ésta es finalizada (Figura 4.4.3 y Figura 4.4.4). Los segmentos de llamada están etiquetados con un par de identificadores. Cada extremo del segmento de la llamada asigna el

identificador fuente o destino durante el proceso de creación del segmento.

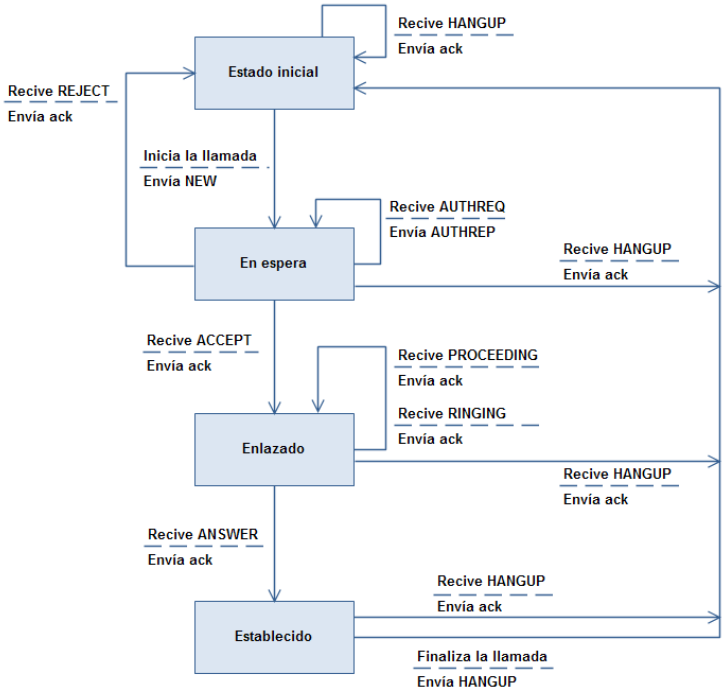


Figura 4.4.3: Inicio de llamada IAX.

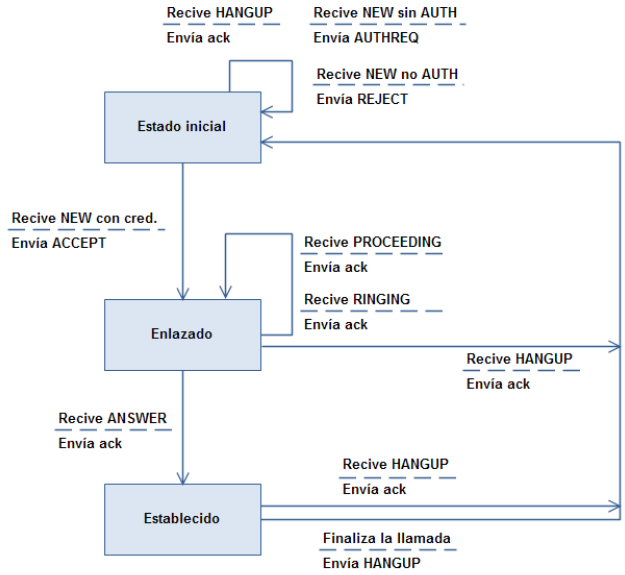


Figura 4.4.4: Finalización de la llamada IAX.

4.4.2.3. Control de llamadas

Los mensajes de control de llamada proveen funciones comunes de señalización extremo a extremo hacia otros protocolos de control telefónicos. Entre los mensajes se incluyen RINGING, ANSWER, BUSY y PROCEEDING. Estos mensajes deben ser enviados únicamente después que un segmento de llamada haya sido aceptado.

En respuesta a un intercambio de mensajes comenzado con NEW, generalmente, el primer mensaje de control de llamada es RINGING; sin embargo, un mensaje PROCEEDING puede proceder a éste, o la llamada puede proceder directamente con un mensaje ANSWER. Si la llamada es contestada, se enviará un mensaje ANSWER. Otras posibilidades incluyen una notificación de ocupado (BUSY), o si no puede ser alcanzado el otro participante de la llamada, ésta será finalizada con un mensaje HANGUP a nivel de enlace junto con el código correspondiente de la causa.

Si el enlace fue comenzado con un mensaje DIAL, la secuencia de mensajes consiste de un PROCEEDING opcional, luego un RINGING también opcional y finalmente un ANSWER o BUSY. En cualquier momento puede ocurrir el envío de un mensaje HANGUP a nivel de enlace. Un par IAX que recibe un mensaje de control que no reconoce debe responder con un mensaje UNSUPPORT.

Además de las funciones comunes, se han desarrollado una gran variedad de extensiones privadas que utilizan estos mensajes de control IAX para pasar datos específicos de aplicaciones a través de los enlaces de control IAX. Un ejemplo de estas extensiones es una aplicación que controla los transceptores de los radio aficionados.

4.4.2.4. Operaciones en el enlace durante la llamada (Mid-Call Link Operations)

Los mensajes de ésta categoría son opcionales y se emiten cuando una llamada es activada. Algunos de los mensajes son:

- Mensaje de solicitud FLASH: Este mensaje es enviado para indicar una característica durante la llamada. Su interpretación es dependiente del sistema y si no es esperado debería ser ignorado. Generalmente, este mensaje solo es enviado desde adaptadores telefónicos analógicos (ATA) cuando ocurre una pequeña interrupción en el circuito durante una llamada contestada.

- Mensaje de solicitud HOLD: Este mensaje es enviado para lograr que el sistema remoto detenga la transmisión de audio en el canal, y opcionalmente reemplazar el audio con música u otros sonidos. Si el sistema remoto no puede llevar acabo esta solicitud, el mensaje debería ser ignorado.
  El mensaje HOLD solo debería ser enviado en las llamadas IAX que han sido iniciadas con el mensaje DIAL.

- Mensaje de solicitud UNHOLD: Este mensaje es enviado para lograr que el sistema remoto vuelva a transmitir el audio en el canal. Si el sistema remoto no puede llevar acabo esta solicitud, el mensaje debería ser ignorado.
  El mensaje UNHOLD solo debería ser enviado en las llamadas IAX luego de un mensaje HOLD.

- Mensaje de solicitud QUELCH: Este mensaje es enviado para lograr que el par remoto detenga la transmisión de audio en el canal. Éste puede reemplazar el audio enviado con música u otros sonidos. Si el sistema remoto no puede llevar acabo esta solicitud, el mensaje debería ser ignorado.
  Este mensaje solo debe enviarse en llamadas IAX luego de que un mensaje ACCEPT es recibido o enviado. Solo debería ser utilizado en llamadas que han sido iniciadas usando el mensaje NEW.

- Mensaje de solicitud UNQUELCH: Este mensaje es enviado para lograr que el sistema remoto vuelva a transmitir audio en el canal. Si brevemente había reemplazado el audio con música u otro sonido, éste debe descontinuar la transmisión de este sonido inmediatamente. Si el sistema remoto no puede llevar acabo esta solicitud, el mensaje debería ser ignorado. El mensaje UNQUELCH solo debería ser enviado en las llamadas IAX luego de un mensaje QUELCH.

- Mensaje de solicitud de TRANSFER: Este mensaje es enviado para lograr que el par receptor reinicie la llamada usando algún otro número especificado. El receptor debió haber iniciado la llamada en el segmento y el comportamiento que la llamada debe tomar no está especificado. Luego de procesar el mensaje, debería enviarse un mensaje HANGUP y finalizarse el segmento de llamada.

Cuando es enviado un mensaje TRANSFER, el nuevo número al cual la llamada es transferida debe incluirse en el CALLED_NUMBER IE (extensión de información CALLED_NUMBER) y además puede incluirse un CALLED_CONTEXT IE. El segmento de llamada no debe ser utilizado para nada más y puede ser finalizado.

4.4.2.5. Optimización del camino de la llamada (opcional)

Cuando un par está manejando una llamada entre otros dos pares IAX y éste ya no tiene necesidad de seguir monitoreando el proceso, contenido, o duración de la llamada, puede removerse a sí mismo de la llamada indicándole a los otros dos pares que se comuniquen directamente entre ellos. Esta optimización del camino de la llamada, o "transferencia supervisada", es lograda de tal manera que asegura que la llamada no se perderá en el proceso; el par iniciador no abandona el control del proceso hasta que ha confirmado que los otros dos pares se están comunicando. Las partes involucradas en la llamada no están en conocimiento de esta operación, debido a que es únicamente una operación de red.

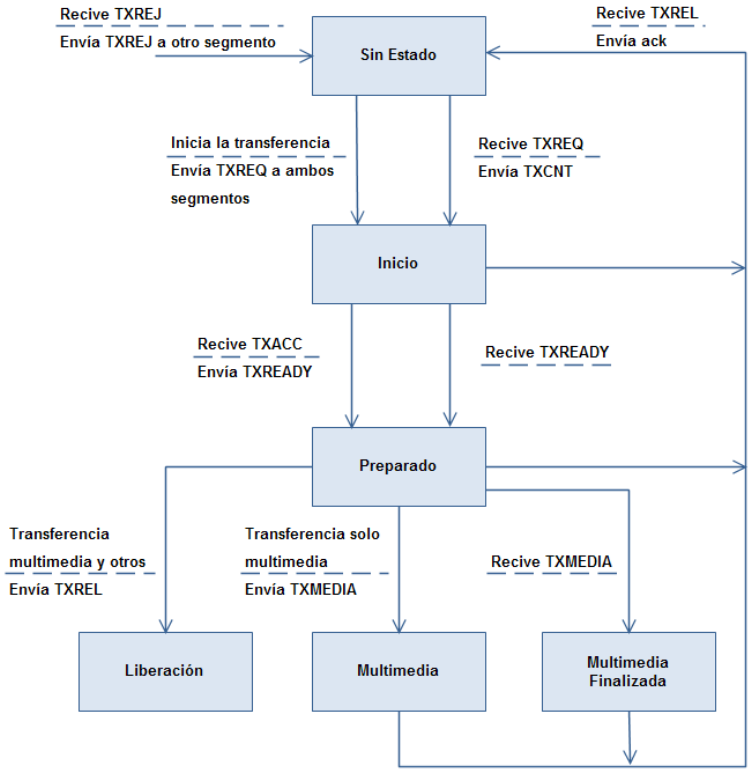


Figura 4.4.5: Optimización de la llamada.

Cuando un par inicia este procedimiento (Figura 4.4.5), ambos segmentos de llamada deben estar en estado UP, es decir, deben haber enviado o recibido el mensaje ACCEPT para ese segmento de llamada. Para empezar, este par envía un mensaje TXREQ con las direcciones e información de los otros pares remotos a cada uno de sus vecinos. Si estos son capaces de llevar a cabo este procedimiento, comienzan transmitiendo toda la información del canal tanto al par iniciador como al nuevo par remoto. Estos también envían un mensaje TXCNT indicando el conteo de paquetes para el segmento de llamada hacia el nuevo par remoto. Cada mensaje TXCNT es confirmado con un mensaje TXACC como respuesta. Los pares responden enviando un mensaje TXREADY al iniciador indicando que han confirmado el nuevo camino de comunicación. Una vez que todos los pares remotos han enviado al iniciador un mensaje TXREADY, la transferencia es completada satisfactoriamente y el iniciador responde con un TXREL y termina su participación en la llamada. Si durante el proceso de transferencia los dos pares remotos no pueden comunicarse, ellos envían un mensaje TXREJ al iniciador.

4.4.2.6. Abandono de llamada (Call Tear Down)

Los mensajes utilizados para finalizar una llamada varían dependiendo del proceso en particular en que se encuentre la llamada en ese momento. Los mensajes para terminar una llamada son: HANGUP, REJECT, TRANSFER, TXREADY.

4.4.2.7. Monitoreo de red

El protocolo IAX tiene varias herramientas para determinar la carga de la red. Utiliza el mensaje POKE para monitorear cuándo un par remoto es alcanzable y el mensaje LAGRQ para medir la calidad del segmento de llamada actual incluyendo el tiempo utilizado para procesar un frame.

4.4.2.8. Marcado digital (opcional)

El soporte para el marcado digital es una característica opcional del protocolo IAX diseñado para soportar dispositivos que no mantienen sus propios planes de marcado (dial plan), por ejemplo, adaptadores telefónicos analógicos (ATA). Los mensajes para marcado digital son: DPREQ, DPREP y DIAL.

4.4.2.9. Varios

Existen algunos mensajes que no pueden ser agrupados para ser clasificados en una categoría específica. Estos mensajes son:

***ACK*:** Confirma la recepción de un mensaje IAX. Es utilizado ante la recepción de un Full Frame que no tiene definido ningún protocolo de respuesta.

***INVAL*:** Este mensaje es enviado como una respuesta a un mensaje recibido que no es válido. Esto ocurre cuando un par IAX envía un mensaje en una llamada luego de que el par remoto ha terminado su parte de la llamada. Luego de recibir este mensaje, el par debe terminar también su parte de la llamada.

***VNAK*:** Es enviado cuando un mensaje es recibido fuera de orden, particularmente cuando un Mini Frame es recibido antes que el primer frame de voz completo en una llamada. Es una solicitud de retransmisión de mensajes descartados. Un mensaje es considerado fuera de secuencia cuando el número de secuencia recibido es diferente al esperado. Luego de la recepción de un VNAK, el par debe retransmitir todos los frames con un número de secuencia más grande que el número de secuencia del VNAK.

***MWI*:** Es utilizado para indicar al par remoto que tiene uno o más mensajes esperando. Este puede incluir una extensión de información (IE) para especificar cuantos mensajes tiene en espera.

***UNSUPPORT*:** Este mensaje es enviado en respuesta a un mensaje que no es soportado por un par IAX. Esto ocurre cuando se recibe un comando IAX con una subclase no reconocida o no soportada.

4.4.2.10. Mensajes multimedia

IAX soporta una gran variedad de tipos multimedia que son transportados por el mismo puerto UDP junto con los mensajes IAX. La voz y el video son únicos en el sentido que utilizan dos tipos de codificaciones distintas, cada una con diferentes procedimientos de soporte. Los Mini Frames abreviados son utilizados normalmente para audio y video; sin embargo, cada vez que la estampilla de tiempo es un múltiplo de 32768 (0x8000 HEX) un Full Frame estándar debe ser enviado. Este enfoque facilita la eficiencia y la confiabilidad enviando paquetes comprimidos, sin

entrega garantizada, la mayor parte del tiempo mientras fuerza periódicamente intercambios confiables con el par. Si la comunicación falla, se invoca el procedimiento de finalización de llamada. Ante la recepción de cualquier mensaje multimedia, excepto los Mini Frames de video y audio abreviados, debe ser enviado un ACK.

Los mensajes multimedia son: DTMF, Voice Media, Video Media, Text Media, Image Media, HTML Media y Confort Noise Media.

4.4.3. Transmisión de mensajes

El protocolo IAX trabaja sobre el protocolo UDP y utiliza un protocolo del nivel de aplicación para proveer transporte confiable cuando resulte necesario.

Existen dos formatos de mensajes: Confiables (Full Frames), y no confiables (Mini o Meta Frames). Todos los mensajes son confiables a excepción de algunos mensajes de voz y video. Los mensajes confiables son transportados por un esquema que mantiene el conteo de mensajes y estampillas de tiempo para ambos participantes involucrados en la llamada. Este conteo se realiza por llamada. Además, cada par mantiene un temporizador para todos los mensajes confiables y debe periódicamente retransmitirlos hasta que se reciba la confirmación de recepción de dichos mensajes o se exceda el límite de reintentos.

Cuando comienza una llamada, los números de secuencia de los mensajes salientes y entrantes deben ser establecidos en cero. Cada mensaje confiable que es enviado incrementa el contador en uno excepto por los mensajes ACK, INVAL, TXCNT, TXACC y VNAK. El mensaje incluye el contador de mensajes salientes y el mensaje de ingreso con el número de secuencia más alto que ha sido recibido. Además, contiene una estampilla de tiempo que representa el número de milisegundos desde que se inició la llamada. O en el caso de que ciertas redes cronometren los mensajes, este contiene una copia de la estampilla de tiempo que recibió.

Cuando se recibe cualquier mensaje, se deben verificar las estampillas de tiempo para asegurar que estos mensajes han llegado en orden. Si un mensaje es recibido fuera de orden, este debe ser ignorado y se debe enviar un mensaje de VNAK para sincronizar los pares. Si el mensaje es un mensaje confiable, el contador de mensajes entrantes debe ser utilizado para confirmar la llegada

de todos los mensajes que han sido enviados hasta el número de secuencia de dicho mensaje.

Si no se recibe confirmación después de un número lógicamente configurado de reintentos (por defecto 4), el segmento de llamada debería ser considerado inutilizable y la llamada debería ser finalizada sin ninguna otra interacción en dicho segmento.

4.4.3.1. Multiplexación de llamadas o Trunking

IAX permite a múltiples intercambios multimedia entre los mismos dos pares ser multiplexados dentro de una única llamada principal mediante la unión del payload multimedia en un paquete combinado. Esto disminuye el ancho de banda usado debido a que existen menos paquetes transmitidos. Este proceso puede ocurrir en cualquiera de las direcciones del intercambio de mensajes. Una llamada principal consiste de un encabezado principal (trunk header) y una o más llamadas IAX agrupadas. El mensaje principal contiene una estampilla de tiempo especificando el tiempo de transmisión del Frame principal. Los datos de audio de las llamadas agrupadas son encapsulados en el Frame principal después del encabezado. Cada llamada encapsulada consiste de dos octetos especificando el número fuente de la llamada, dos octetos especificando la longitud en octetos de los datos multimedia, y los datos multimedia en sí. IAX permite también transmitir las estampillas de tiempo de cada Mini Frame encapsulado, de forma que la información precisa de cronometrado puede ser usada para los buffers de variación. Una bandera en el encabezado especifica cuando un Mini Frame encapsulado retiene su estampilla de tiempo original. Si los pares no retienen sus estampillas, deben asumir que la estampilla de tiempo es la recibida en el encabezado principal al momento del arribo.

4.4.3.2. Temporizadores

El protocolo IAX utiliza diversos temporizadores para distintos motivos, entre los más importantes se pueden destacar al Temporizador de retransmisión y el Temporizador de periodo de registración. El Temporizador de retransmisión es utilizado para medir cuanto debe esperar por la confirmación de un mensaje. Este temporizador se inicia con el doble del valor del RTT (Round-Trip Time) del último comando PING/PONG. Si es necesaria una retransmisión, este valor es incrementado hasta alcanzar un tope máximo fijado por defecto en 10 segundos. El Temporizador de periodo de registración se encarga de especificar por cuánto tiempo es válida una registración. Es responsabilidad del cliente renovar la misma antes que el temporizador alcance el periodo de

expiración. Las registraciones deberían ser renovadas a intervalos aleatorios para prevenir congestión en la red. Un servidor registrar debe monitorear este periodo de tiempo e invalidar la registración si el cliente no la renovó antes que finalice el periodo.

4.4.3.3. Consideraciones sobre NAT

IAX está preparado para operar satisfactoriamente detrás de NAT debido a su enfoque de puerto único. Este enfoque elimina cualquier retraso en el comienzo de una llamada de flujo multimedia mientras el Gateway NAT establece una asociación bidireccional de puertos. La implementación de un único servidor IAX detrás de un Gateway NAT requiere de poco esfuerzo. Si el servidor actúa como un servidor registrar, el puerto UDP de IAX en el Gateway debe ser re-direccionado al servidor. Si el servidor actúa como un cliente registrante, el Temporizador de periodo de registración por defecto de 60 segundos debería ser suficiente para mantener una asociación de puertos en el Gateway NAT; aunque es preferible una asociación de puertos estática.

Cuando la implementación incluye múltiples servidores detrás de un único Gateway NAT, en la mayoría de los casos los Gateways requieren que cada servidor IAX utilice un puerto UDP distinto. Aunque puede haber algunas implementaciones NAT que reconozcan cuando múltiples dispositivos utilizan el mismo puerto privado y lo administran correctamente.

4.4.3.4. Encriptación

IAX soporta encriptación de llamadas usando la clave simétrica AES (Advanced Encryption Standard). IAX encripta llamada a llamada empezando con un mensaje NEW en texto plano indicando, junto con otros parámetros, que la llamada debe encriptarse. Esta indicación es dada a través de la utilización de la extensión de información (IE) ENCRYPTION en el mensaje NEW. Si el par contactado soporta encriptación, éste responderá con un mensaje AUTHREQ en texto plano que también incluye la extensión de información ENCRYPTION. Todos los mensajes siguientes en la llamada serán encriptados. Si el par contactado no soporta encriptación, el mensaje AUTHREQ enviado no debe incluir dicha IE y el par que inicio la llamada debe finalizar la misma o continuarla sin encriptación.

4.4.4. Estructura de los mensajes

IAX distingue entre varios tipos de frames. Existen tres categorías principales que son manipuladas por el protocolo, estas son Mini, Full y Meta frames. IAX también transporta frames encriptados pero estos no son considerados tipos de frames IAX. A continuación se detallarán las principales categorías de frames IAX y las diferencias entre ellos en cuanto a su uso y estructura.

4.4.4.1. Full Frames

Los Full Frames pueden enviar datos de señalización o de multimedia. Generalmente, son utilizados para el control de la iniciación, configuración y terminación de una llamada IAX, pero pueden ser utilizados también para transportar flujos de datos, aunque generalmente esto no es óptimo.

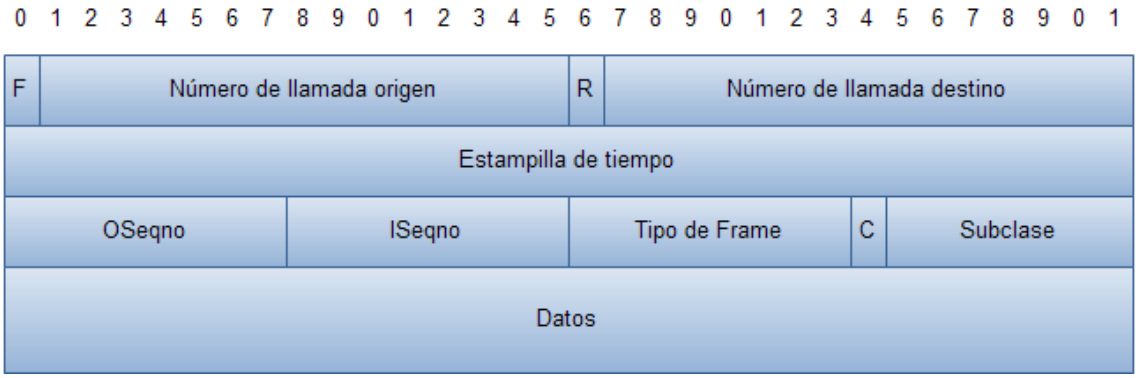


Figura 4.4.6: Encabezado Full Frame.

Los Full Frames entran en la categoría de mensajes confiables, por lo tanto requieren una confirmación de recibo inmediatamente al ser recibidos. Esta confirmación puede ser explícita a través de un mensaje ACK o implícita a través de la respuesta correspondiente a dicho Frame. El encabezado estándar de un Full Frame (Figura 4.4.6) posee una longitud de 12 octetos y la descripción de los campos que lo componen son:

- Bit “F”: Este bit indica si el frame es del tipo Full Frame cuando su valor está establecido en uno.

- Número de llamada origen: Este valor de 15 bits especifica el número de llamada que el cliente emisor utiliza para identificar a la misma. Este número en una llamada activa no debe estar disponible para otra llamada en el mismo cliente. El número de llamada puede ser reusado una vez que la llamada se vuelve inactiva.

- Bit “R”: Indica si el frame está siendo retransmitido o no. Si su valor está establecido en cero, significa que el frame está siendo transmitido por primera vez. Si su valor está establecido en uno, indica que el frame está siendo retransmitido. IAX no especifica el período de tiempo de retransmisión, esto queda a cargo del implementador.

- Número de llamada destino: Especifica el número de llamada que el cliente emisor utiliza para referenciar la llamada en el par remoto. Este número es el mismo del campo *número de llamada origen* del par remoto. El número de llamada destino identifica unívocamente una llamada en el par remoto.

- Estampilla de tiempo: Este campo contiene una estampilla de tiempo de 32 bits mantenida por un par IAX para una llamada dada. La estampilla de tiempo es una representación incremental de los milisegundos transcurridos desde la primera transmisión de la llamada.

- OSeqno: Este campo de 8 bits es el número de secuencia del flujo de salida. Al inicio de la llamada este valor es cero, y se incrementa a medida que son enviados los Full Frames. Si el valor de esta variable es desbordado, es reiniciado a cero nuevamente.

- ISeqno: Este campo de 8 bits es el número de secuencia del flujo de entrada. Al inicio de la llamada este valor es cero, y se incrementa a medida que son recibidos los Full Frames. Si el valor de esta variable es desbordado, es reiniciado a cero nuevamente.

- Tipo de Frame: Identifica el tipo de mensaje transmitido por el frame. Los tipos de frames pueden ser DTMF, voz, video, control, nulo, IAX, texto, imagen, HTML y Sonido de confort.

- Bit “C”: Este bit determina cómo los 7 bits restantes del campo Subclase están codificados. Si el bit “C” está establecido en uno, el valor de la Subclase es interpretado como una potencia de dos, en caso contrario es interpretado como un número entero positivo de 7 bits.

4.4.4.2. Mini Frames

Los Mini Frames son llamados de esta manera debido a que su encabezado está formado por sólo cuatro octetos. Estos frames no transportan datos de control ni de señalización, son utilizados únicamente para transportar flujos multimedia en una llamada IAX ya establecida. Son enviados como no confiables, debido que las llamadas VoIP generalmente pueden perder varios frames sin

sufrir una degradación significativa en la calidad de la llamada mientras que el costo de procesamiento para asegurar confiablidad incrementa los requerimientos de ancho de banda y disminuye la eficiencia. Más aún, dado que las llamadas de voz son generalmente enviadas en tiempo real, los frames perdidos resultan ser antiguos para ser reintegrados en el flujo de audio al momento que estos pueden ser retransmitidos. La descripción de su encabezado (Figura 4.4.7) es la siguiente:

- Bit "F": Este bit debe estar establecido en cero.

- Número de llamada origen: Este número es utilizado por el par emisor para identificar la llamada actual.

- Estampilla de tiempo: Este campo de 16 bits está formado por los 16 bits menos significativos de la estampilla de tiempo de 32 bits utilizados en la llamada por el par transmisor. La estampilla de tiempo permite sincronización de frames entrantes por lo cual pueden ser procesados en orden cronológico en lugar de ser procesados en el orden en que llegan. Cuando este valor llega a su límite luego de 65536 segundos se debe enviar un Full Frame para notificar al par remoto que la estampilla de tiempo ha sido restablecida. La llamada debe continuar transmitiendo Mini Frames comenzado con la estampilla de tiempo cero aún si no se ha recibido la confirmación de sincronización.

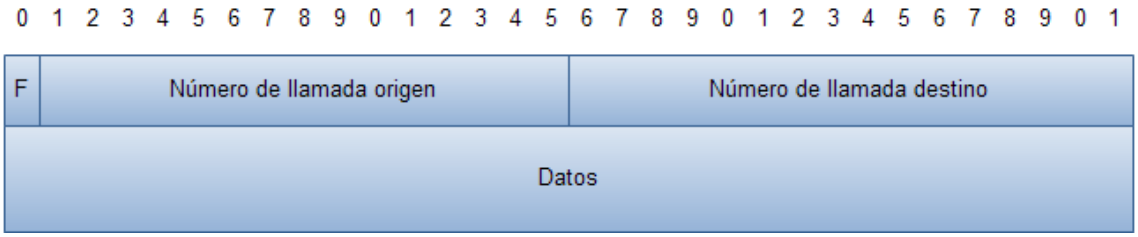


Figura 4.4.7: Encabezado Mini Frame.

4.4.4.3. Meta frames

Los meta frames sirven para satisfacer dos propósitos. Los meta frames de video permiten la transmisión de flujos de video con un encabezado optimizado, y su propósito es similar al de los mini frames de voz. Los meta frames troncales son utilizados para unir múltiples flujos multimedia IAX entre dos pares en un solo encabezado, y de esta forma reducir el consumo de ancho de banda. Los campos que componen la estructura del encabezado de los meta frames de video es la

siguiente:

- Bit “F”: El bit indica si el frame es de tipo Full Frame, para los meta frame este bit siempre debe ser cero.

- Indicador meta: Es un campo de 15 bits con valor igual a cero, utilizado para indicar que el frame es del tipo meta frame. Los meta frames son identificables debido a que los primeros 16 bits siempre son cero, mientras que en Full Frames y Mini Frames siempre van a tener el bit F establecido o algún valor distinto de cero en el número de llamada origen.

- Bit “V”: Este bit debe estar establecido en uno para indicar que se trata de un Meta Frame de video.

- Número de llamada origen: Es utilizado por el par transmisor para identificar a la llamada de video.

- Estampilla de tiempo: Este campo de 16 bits está formado por los 16 bits menos significativos de la estampilla de tiempo de 32 bits utilizados en la llamada por el par transmisor. Cuando este valor llega a su límite luego de 65536 segundos se debe enviar un Full Frame para notificar al par remoto que la estampilla de tiempo ha sido restablecida.

4.4.4.4. Meta Frame Troncal

IAX soporta de forma nativa dos métodos para agrupar múltiples flujos multimedia entre dos pares dentro de una misma asociación (llamada principal). El primer método envía un meta encabezado estándar, junto con una única estampilla de tiempo de 32 bits que describen el momento de transmisión del frame troncal. A continuación de la estampilla se encuentra uno o más frames multimedia que consisten del número de la llamada y la longitud en octetos del flujo de datos incluidos en el frame.

El segundo método es muy similar al primero. Este envía un meta encabezado estándar, incluyendo la estampilla de tiempo de 32 bits. Pero los frames multimedia incluidos, en realidad, se tratan de mini frames completos, incluyendo la estampilla de tiempo de 16 bits por cada llamada.

El primer método utiliza menos ancho de banda (dos octetos menos por cada llamada en la asociación) mientras que el segundo método mantiene las estampillas de tiempo para cada llamada, por lo tanto el buffer de jitter puede utilizar la estampilla de tiempo real asociada con la llamada, en lugar de la estampilla de tiempo que utiliza la asociación entera que es menos precisa.

La descripción de los campos es la siguiente:

- Bit “F”: Este bit debe ser establecido en cero, dado que no se trata de un Full Frame.

- Indicador Meta: Es un campo de 15 bits con valor igual a cero, utilizado para indicar que el frame es del tipo Meta Frame. Los meta frames son identificables debido a que los primeros 16 bits siempre son cero, mientras que en Full Frames y Mini Frames siempre van a tener el bit F establecido o algún valor distinto de cero en el número de llamada origen.

- Bit “V”: Este bit debe estar establecido en cero, dado que no se trata de un Meta Frame de video.

- Comando Meta: Este campo de 7 bits identifica cuando el Meta Frame es una asociación. Si está establecido en uno significa que el Frame es un Meta Frame troncal. El resto de los valores posibles están reservados para usos futuros.

- Datos de comando: Este campo de 8 bits especifica banderas para opciones que se aplican a llamadas multiplexadas. El bit menos significativo del campo se denomina bandera "trunk time-stamps". Si el valor de este bit está establecido en cero, indica que las llamadas en la asociación no incluyen sus estampillas de tiempo individuales. Si el valor está establecido en uno, indica que cada llamada incluye su propia estampilla de tiempo. Todos los bits restantes en el campo están reservados para usos futuros.

- Estampilla de tiempo: Este campo de 32 bits representa el tiempo real de la transmisión del frame troncal, el cual es distinto al tiempo de las llamadas incluidas en él.

Luego de la estampilla de tiempo de 32 bits sigue una o más llamadas multiplexadas. Si la bandera “trunk time-stamps” está establecida en cero, cada una de estas entradas consiste de dos octetos que especifican el número de llamada origen, dos octetos que especifican la longitud en octetos de los datos multimedia, y luego los datos multimedia. Si dicha bandera está establecida en uno,

cada una de las entradas consiste de dos octetos especificando la longitud en octetos de los datos multimedia, y luego un Mini Frame (dos octetos especificando el número de llamada origen, dos octetos especificando la estampilla de tiempo y los datos multimedia). En la Figura 4.4.8se ilustra la estructura de los Meta Frames Troncales.

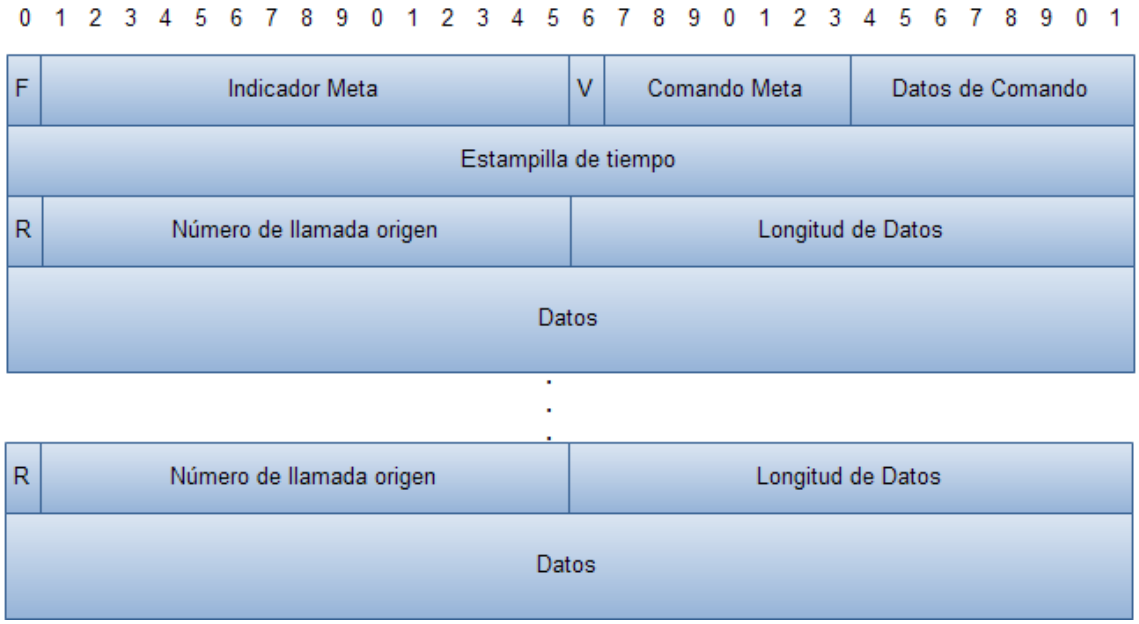


Figura 4.4.8: Meta Frame Troncal.

4.5. Resumen

Para lograr la implementación de una central telefónica IP es necesario entender cómo es que se llevan a cabo las comunicaciones entre los distintos elementos participantes. Es por esto que la investigación sobre los protocolos de señalización abordados por este capítulo es de importancia a un nivel conceptual.

Un protocolo de señalización se encarga de la parametrización de la comunicación, un aspecto fundamental ya que a través de ésta se negocian las opciones que se utilizarán para llevar a cabo una llamada. Este, además, se encarga de mantener un control de la llamada y de la renegociación de los parámetros durante la misma si fuese necesario y de la finalización de las sesiones. Dentro de sus funciones principales se encuentran la registración y ubicación de usuarios, y el enrutamiento de las llamadas. En este capítulo se describió a los protocolos H.323, SIP e IAX,

siendo los últimos dos los principales protocolos intervinientes en las comunicaciones VoIP.

SIP (Session Initiation Protocol) es un protocolo de señalización de propósito general que es fuertemente utilizado en telefonía IP, generalmente es utilizado para la comunicación entre los clientes y los servidores. Sólo se encarga de la señalización de la llamada, mientras que para la transmisión de la misma se utilizan los protocolos RTP (*Real-Time Transport Protocol*) y RTCP (*Real-Time Transport Control Protocol*).

El protocolo IAX (**I**nter-**A**steriske**X**change) suele ser utilizado en la señalización de la comunicación entre los servidores VoIP, debido a que fue diseñado para solucionar los problemas que se presentan con NAT en SIP, ya que no requiere de protocolos adicionales para el transporte de los datos multimedia. Por otra parte, es independiente del protocolo de red y por lo tanto se adapta fácilmente sobre el protocolo de red que se utilice, ya sea IPv4 o IPv6, mientras que SIP debe ser modificado para lograr esto.

Se debe destacar la importancia que tienen estos protocolos ya que son los que permiten tanto establecer las comunicaciones telefónicas entre los extremos como así también todo el intercambio de información de control y señalización. Además, es importante el uso de estos protocolos en particular debido a que son estándares y abiertos, facilitando la interconexión con otros sistemas ya existentes.

Capítulo 5: Implementación de Sistemas Distribuidos y Alta disponibilidad en Linux

5.1. Introducción

La evolución de la informática y las redes de comunicación crearon y permiten una necesidad de consumo continua, sin pérdidas de datos, de tiempo, sin fallas y sin caídas del servicio. Un ejemplo de esto son las transacciones bancarias o de la bolsa de comercio mundial, transacciones que deben estar disponibles en cualquier momento y en cualquier lugar para brindar un servicio de primera línea y sin perder ni un centavo. Las perdidas monetarias que se pueden producir por la caída de estos servicios, suponga durante una hora, son inmensurables.

La mejor forma de afrontar estos requerimientos es mediante la implementación de una infraestructura distribuida y de alta disponibilidad. El objetivo de ésta tesis no es tan crítica como el ejemplo dado, pero éste solo fue a modo de ilustrar la importancia de estos dos conceptos (los sistemas distribuidos y la alta disponibilidad). Esta implementación puede realizarse utilizando una combinación de soluciones software y soluciones hardware. Las primeras son herramientas a nivel de aplicación que brindan capacidades de comunicación entre nodos y administración de recursos y servicios, mientras que las segundas son utilizadas de apoyo a las primeras tales como la replicación de dispositivos, dispositivos de respaldo, dispositivos de almacenamiento compartido, entre otros.

Una de las principales ventajas que presenta la implementación de una arquitectura de clúster es el balanceo de carga de datos, que permite mantener una carga de procesamiento equivalente entre los nodos que lo conforman aumentando significativamente el tiempo de respuesta, tiempo de procesamiento, como también el volumen de información procesada. Para realizar esta tarea es necesario una infraestructura que cuente con un nodo central capaz de distribuir las peticiones entre los nodos participantes del clúster y mantener información de las conexiones activas. Una de las herramientas capaces de llevar a cabo esta arquitectura es Linux Virtual Server (LVS) capaz de brindar una visión de trabajo sobre un único servidor. Esta solución por software puede ser

acompañada por otras tales como Heartbeat y Pacemaker que permiten dotar de alta disponibilidad a los servicios, que pueden también estar distribuidos.

Además, es importante llevar a cabo tareas de monitoreo que permitan conocer el estado global del sistema pudiendo detectar así las fallas que puedan ocurrir en los servicios o en los nodos con el fin de re-establecerlos lo antes posible sin que el usuario pueda notarlo. Esta operación es un concepto implícito dentro de los conceptos de alta disponibilidad y sistemas distribuidos.

Por último, los conceptos mencionados pueden ser aplicados para implementar distintas arquitecturas que proveen alta disponibilidad en un clúster conformado por servidores Asterisk, brindando telefonía IP con alta disponibilidad, que asegure servicio continuo, con servidores que pueden estar distribuidos geográficamente.

5.2. Sistemas distribuidos

Como se menciona en Distributed Systems - Principles and Paradigms (Andrew S.Tanenbaum, 2006), la industria de la computación ha sido la industria con mayores cambios y mejoras en los todos aspectos. Desde las primeras computadoras de gran tamaño y un costo elevadísimo, capaces de procesar una instrucción por segundo, a las computadoras de hoy en día, pequeñas, baratas y capaces de procesar un billón de instrucciones por segundo. No hay dudas, que la revolución que presenta esta industria es única y sin precedentes.

Se puede hacer hincapié en dos avances tecnológicos para darle origen a estos hechos. El primero, fue el desarrollo de poderosos microprocesadores. Desde las CPUs de 8-bit, pasando a los de 16, 32 y finalmente a los de 64-bit. Este crecimiento hizo que una computadora moderna tenga el poder computacional de una mainframe antigua pero a una fracción del precio. El segundo avance a mencionar, fue la invención de las redes de computadoras de alta velocidad. Las redes de área local (o LANs) permiten que cientos de computadoras dentro de un edificio sean conectadas de modo que pequeñas cantidades de información puedan ser transferidas entre ellas en pocos milisegundos o menos. Las redes de área extensa (o WANs) permiten que millones de computadoras alrededor del planeta puedan conectarse para transmitir grandes volúmenes de información a grandes velocidades.

El resultado de estas tecnologías es que hoy en día no sólo es posible, sino también fácil, armar sistemas de computación compuestos por un gran número de computadoras conectadas por una red de alta velocidad logrando así un poder computacional elevadísimo a un menor costo. Usualmente, éstos son llamados redes de computadoras o sistemas distribuidos, en contraste a los anteriores sistemas centralizados (o sistemas de un único procesador) conformados de una sola computadora.

Esta ha sido una breve introducción al surgimiento de los sistemas distribuidos, a continuación citaremos la definición propuesta de distintos autores:

- **Coulouris:** "Sistema de computadoras interconectadas que comunican y coordinan sus acciones solamente mediante el pasaje de mensajes".

- **Tanenbaum:** "Una colección de computadoras independientes que aparentan ser ante los usuarios un sistema coherente único".

- **Lamport:** "Sabemos que tenemos un Sistema Distribuido cuando ante la caída de una computadora nunca escucharemos acerca de paradas que impidan la finalización de un trabajo".

- **Blair y Stefani:** "Un Sistema Distribuido es un sistema diseñado para soportar el desarrollo de aplicaciones y servicios que puedan aprovechar una arquitectura física consistente en elementos de procesamiento autónomo múltiple que no comparten memoria primaria pero cooperan entre sí enviando mensajes asincrónicos a través de la red".

De estas definiciones podemos desprender las siguientes características que deberían presentarse un sistema distribuido (SD):

- Sistema débilmente acoplado.
- No existe un reloj común.
- Dispositivos de E/S asociados a cada procesador.
- Fallos independientes de componentes del SD.

- Componentes heterogéneos.
- Uso de interfaz software que oculta la complejidad hardware de un SD.
- Visión de sistema único (Single System Image).

Sin lugar a dudas, los avances tecnológicos que dan respuesta a nuevas necesidades traen consigo ventajas y desventajas que harán de bisagra para determinar si es conveniente su uso y bajo qué circunstancias. Es por esto que es importante mencionarlas, entre las ventajas que podemos discernir encontramos:

- **Economía** (buena relación rendimiento / costo).
- **Alto rendimiento** (procesamiento paralelo).
- **Soporte de aplicaciones inherentemente distribuidas** (por ejemplo, empresa distribuida geográficamente).
- **Capacidad de crecimiento** (Escalabilidad).
- **Fiabilidad y disponibilidad** (Tolerancia a fallas).
- **Carácter abierto y heterogéneo** (Estándares de interoperabilidad).
- **Colaboración** (compartir recursos y datos).

Entre las desventajas podemos destacar:

- **Necesidad de un nuevo tipo de software** (más complejo de lo normal, debe sortear con los problemas de comunicación, ubicación y poseer interoperabilidad).

- **Comunicación** (pérdida de mensajes y saturación, la latencia puede provocar que al recibir un dato ya esté obsoleto).

- **Seguridad y confidencialidad** (nuevos desafíos a superar y mayor cantidad de elementos a proteger).

Es importante notar que en un sistema centralizado, toda la información se encuentra almacenada en un solo sitio, en cambio en un sistema distribuido cada nodo posee información parcial de la situación lo que dificulta mantener la consistencia en el sistema. Más allá de las desventajas

anteriores, la limitación más importante que poseen los sistemas distribuidos es la ausencia de un reloj y una memoria global. Esto hace que lograr un estado único, global y consistente para que los usuarios tengan una visión única del sistema es la tarea más difícil de alcanzar en los sistemas distribuidos.

5.2.1. Tipos de Sistemas Distribuidos

Una importante clase de sistemas distribuidos es el utilizado para las tareas de computación de alto rendimiento. En términos generales, se puede hacer distinción entre dos subgrupos. Los clústers, donde el hardware subyacente consiste de una colección PCs o estaciones de trabajo similares, que se encuentran en áreas estrechamente cercanas conectadas por redes LAN de alta velocidad. Además, cada nodo corre con el mismo sistema operativo. Esta homogeneidad las hace más sencillas de implementar.

En cambio, el subgrupo de los grid consiste en una construcción federal de sistemas de computadora, donde cada uno puede encontrarse en distintos dominios administrativos, que pueden ser bastante diferentes en hardware, software y la tecnología de red implementada.

5.2.1.1. Clústers

Los sistemas de computación clúster se volvieron populares cuando la relación precio / rendimiento de las computadoras y estaciones de trabajo mejoró. En un cierto punto, se volvió atractivo técnica y financieramente construir súper computadoras mediante la conexión de simples computadoras en redes de alta velocidad. En la mayoría de los casos, los clústers son utilizados para correr un programa (de procesamiento intensivo) en paralelo sobre varias computadoras.

Un ejemplo conocido es el clúster Beowulf basado en Linux, mostrado en la Figura 5.2.1. Consiste de una colección de nodos de cómputo que son controlados y accedidos por medio de un único nodo maestro. El nodo maestro maneja la asignación de nodos para que un programa en particular sea ejecutado en forma paralela, manteniendo una cola de tareas, y provee una interface para los usuarios del sistema. Para esto, el nodo maestro generalmente ejecuta un software intermedio (middleware) necesario para la ejecución y la administración de los programas del clúster, mientras que los nodos de cómputo a menudo solo necesitan un sistema

operativo estándar y nada más. Una parte importante de este middleware está formado por las bibliotecas para la ejecución de programas paralelos, las cuales suelen únicamente ofrecer facilidades para la comunicación basada en mensajes, pero no ofrecen facilidades para manejar fallas, seguridad, etc.

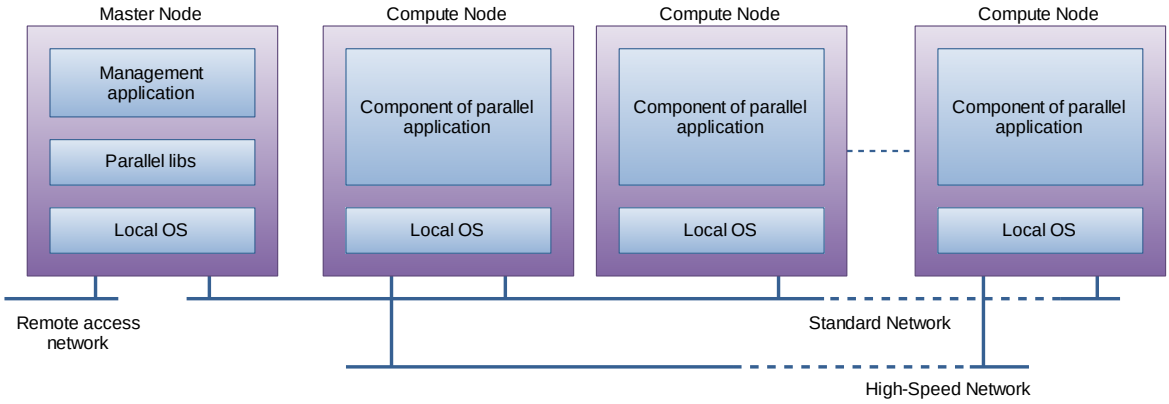


Figura 5.2.1: Ejemplo de sistema de computación clúster.

Una alternativa a esta organización jerárquica, es un enfoque simétrico como el que sigue el sistema MOSIX. MOSIX intenta brindar una imagen única del clúster, es decir, intenta aparentar ante los procesos como una única computadora, ocultando así la distribución del mismo. En los sistemas distribuidos, la transparencia o el ocultamiento de funcionalidades tienen varios aspectos, tal como ocultar el acceso a los recursos, la ubicación, la migración, la concurrencia, etc. Intentar una transparencia total es una tarea imposible, en el caso de MOSIX, el mayor grado de trasparencia es proporcionado al permitir que los procesos puedan migrar de forma dinámica y preventiva entre los nodos que conforman el clúster. La migración de procesos permiten al usuario empezar una aplicación en cualquier nodo, y luego moverse trasparentemente a otros nodos, por ejemplo, para hacer un uso eficiente de los recursos.

5.2.1.2. Grids

Un rasgo característico de clúster es su homogeneidad. En la mayoría de los casos, las computadoras en un clúster son iguales, con el mismo sistema operativo, y conectados a través de la misma red. En contraste, un grid posee un alto grado de heterogeneidad, no se hacen suposiciones sobre el hardware, los sistemas operativos, las redes, dominios administrativos,

políticas de seguridad, etc.

Una cuestión clave en la computación grid es que los recursos de diferentes organizaciones se unen para permitir la colaboración de un grupo de personas o institutos. Esta colaboración se realiza en la forma de una organización virtual, y cada usuario perteneciente a ésta posee los mismos permisos de acceso. Típicamente, los recursos de un grid consisten de servidores, clústers, facilidades de almacenamiento, base de datos, etc. Además, pueden encontrarse dispositivos especiales conectados como los telescopios, sensores, etc.

Dada su naturaleza, la mayor parte del software para la realización del grid gira en torno a facilitar el acceso a los recursos de diferentes dominios administrativos, y sólo a los usuarios y aplicaciones que pertenecen a una organización virtual específica.

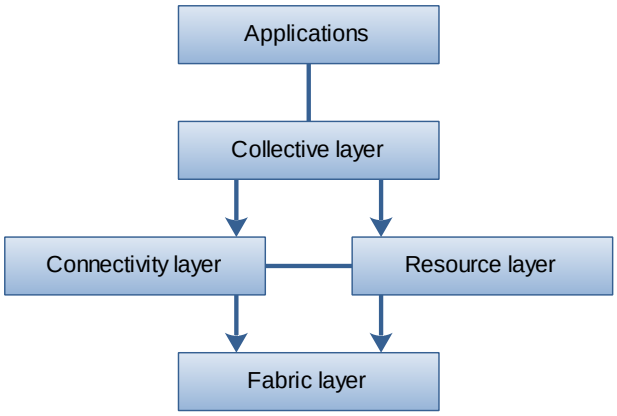


Figura 5.2.2: Ejemplo de sistema de computación grid.

En la Figura 5.2.2, se ilustra una arquitectura posible para los sistemas de computación grid. Esta arquitectura consiste de cuatro capas. La capa más inferior, la “capa de fabricación”, proporciona las interfaces a los recursos locales de un sitio específico, las cuales están diseñadas para permitir el intercambio de recursos dentro de una organización virtual. Por lo general, se ofrecen funciones para consultar el estado y las capacidades de un recurso, junto con las funciones para la administración de dicho recurso.

La capa de conectividad se compone de protocolos de comunicación para soportar las transacciones del grid que se utilizan para acceder a varios recursos. Además, contendrá los protocolos de seguridad para autenticar a usuarios y recursos. Hay que notar que en muchos casos

los usuarios no estarán autenticados, si no los estarán los programas que actúan en el nombre de estos. En este sentido, es importante que la delegación de permisos desde usuarios a programas es una función importante que se necesita que este soportada por la capa de conectividad.

La capa de recursos es responsable de la administración de un recurso. Ésta usa las funciones provistas por la capa de conectividad y llama directamente a las interfaces disponibles en la capa de fabricación. Por ejemplo, esta capa ofrecerá funciones para obtener información de configuración de un recurso especifico, o para realizar operaciones específicas como la creación de procesos o leer datos, etc. De esta forma la capa de recursos es vista como la responsable para el control de acceso, y confiará en la autenticación realizada por parte de la capa de conectividad.

Luego se encuentra la capa de colectividad se encarga de manejar los accesos a múltiples recursos y generalmente consiste de servicios para el descubrimiento de recursos (discovery), asignación y administración de tareas sobre distintos recursos, replicación de datos, y mucho más. A diferencia de la capa de conectividad y la de recursos, las cuales se componen de un grupo relativamente reducido, y estándar de protocolos, la capa de colectividad puede consistir de muchos protocolos distintos para diferentes propósitos, reflejando así el amplio espectro de servicios que éste puede ofrecer a la organización virtual.

Finalmente, la capa de aplicación consiste de aplicaciones que operan dentro de la organización virtual y hacen uso del entorno de computación grid.

Generalmente, las capas de colectividad, conectividad y de recursos son el corazón de la capa de middleware que todo sistema distribuido necesita. Estas capas en conjunto proveen el acceso y la administración de los recursos que pueden estar potencialmente dispersos a lo largo de distintos sitios.

5.3. Alta Disponibilidad

El concepto de Alta disponibilidad (High availability), a diferencia del concepto de sistemas distribuidos, es bastante simple y directo. Se dice que un sistema posee alta disponibilidad cuando se asegura un cierto grado absoluto de continuidad operacional durante un período de medición dado. Disponibilidad se refiere a la habilidad de que usuarios puedan acceder al sistema, someter

nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder o realizar sus tareas se dice que el sistema o servicio no está disponible. Suele asociarse el término tiempo de inactividad (down-time) para definir cuándo el sistema no está disponible.

Se puede dotar con esta característica tanto a servicios como a datos, y la forma de lograrlo puede ser a través de soluciones software y/o hardware. Por lo general, las soluciones hardware son utilizadas para apoyar a las soluciones de tipo software, manteniendo o aumentando el nivel de seguridad y disponibilidad de las mismas. La disponibilidad generalmente se logra con una redundancia de dispositivos o nodos que conforman un sistema. Entre las soluciones hardware también se pueden encontrar Network Attached Storage (NAS), Redundant Array of Independent Disks (RAID), Storage Area Network (SAN), entre otras. Algunas herramientas del tipo software son Heartbeat, KeepAlived, Red Hat Cluster Suite, The High Availability Linux Project, LifeKeeper, etc.

A esta altura, es importante notar que la combinación de sistemas distribuidos y herramientas o soluciones de alta disponibilidad dan una gran potencia para ofrecer a los usuarios una experiencia de calidad única. De esta manera han surgido los clúster de alta disponibilidad, que son un conjunto de dos o más computadoras con una serie de servicios compartidos. Los nodos del clúster se monitorean entre sí continuamente, al detectar la caída o falla de un servicio o nodo, automáticamente éstos son reiniciados en cualquiera de los otros nodos. Cuando el servicio se recupera de su falla, los datos son migrados nuevamente al nodo inicial. Esta recuperación automática proporciona la alta disponibilidad de los servicios ofrecidos, y minimiza la percepción del usuario sobre las fallas del sistema.

Es importante no caer en las confusiones de conceptos relacionados. Frecuentemente, muchas personas confunden disponibilidad con funcionalidad y alta disponibilidad con alto rendimiento. Un sistema puede seguir funcionando y no estar disponible para el acceso del usuario, en este caso el sistema posee funcionalidad pero no disponibilidad. El alto rendimiento es la cualidad de proporcionar capacidades de cálculo mucho mayores de lo común, y alta disponibilidad es la cualidad para garantizar el acceso interrumpido de un servicio determinado.

5.3.1. Configuraciones de redundancia

Las configuraciones de redundancia más comunes y utilizadas son conocidas como Activo/Activo, Activo/Pasivo, N+1, N+M, N:1 y N:N.

- **Activo/Pasivo:** La configuración consta de dos o más nodos, donde uno de ellos toma el papel de nodo principal o maestro y donde estarán corriendo la mayoría de recursos. El resto de nodos ofician de esclavos o de backup y entrarán en funcionamiento cuando el nodo principal falle.

- **Activo/Activo:** Esta configuración también consta de dos o más nodos. Con este tipo de clúster se desarrolla lo que se denomina "balanceo de carga". Generalmente, los recursos están duplicados (clonados) en varios nodos y mediante algún mecanismo se reparte la carga entre cada nodo. Cuando uno de los nodos falla, el resto de los nodos toman el control de los servicios que estaban funcionando en el nodo caído.

- **N+1:** La configuración de redundancia N+1 consta de un componente de recuperación o backup independiente para N nodos que conforman el clúster. Este tipo de configuración tolera únicamente la falla de un nodo por grupo.

- **N+M:** En esta configuración de redundancia existen M nodos independientes de recuperación para N nodos del clúster. Cualquiera de los M nodos puede tomar el lugar de cualquiera de los N nodos del clúster. A diferencia de la configuración anterior, donde solo existe tolerancia a un fallo por grupo, esta configuración tolera M fallos por grupo.

- **N:1 y N:N:** La configuración N:1 puede confundirse fácilmente con la configuración N+1, incluso es común que se la definan como equivalentes. La diferencia claramente está en la etapa de recuperación de fallos. Cuando el nodo que falla es recuperado, en una configuración N+1 el nodo de recuperación se mantiene como el nodo activo de los servicios que tomó, mientras que en la configuración N:1 el nodo recuperado vuelve a tomar nuevamente el control de los servicios. Como puede verse, en la segunda opción

existe 2 interrupciones de las conexiones activas, la primera se produce en el fallo del nodo y la segunda al trasladarse nuevamente los servicios al nodo original. En una configuración N:N existen N nodos de recuperación para N nodos del clúster.

5.4. Soluciones de alta disponibilidad por software en Linux

Heartbeat (Linux-HA, 2010) es la solución no propietaria, desarrollada bajo GNU/Linux por el proyecto Linux-HA, de alta disponibilidad por software más utilizada que permite que servicios críticos se encuentren siempre disponibles a pesar de problemas o fallos que pueden surgir durante su funcionamiento.

Esta herramienta provee una infraestructura de clúster (comunicación y pertenencia) a un conjunto de nodos encargados de llevar adelante el funcionamiento de los servicios críticos. Les permite conocer acerca de la existencia o desaparición de procesos en los nodos pares y de un fácil intercambio de mensajes entre ellos. En función de ser una herramienta poderosamente útil es necesario que sea combinada con un *Administrador de Recursos de Clúster* (Cluster Resource Manager o CRM) capaz de iniciar o detener los servicios a los cuales el clúster brinda alta disponibilidad. El administrador de recursos de clúster más utilizado en forma conjunta con Heartbeat es Pacemaker (ClusterLabs, 2010). La combinación de ambas herramientas permite que un servicio pueda ser migrado de un nodo a otro de forma rápida y transparente a los clientes logrando que el servicio sólo se encuentre interrumpido durante el tiempo en el que Heartbeat detecta el fallo y se lleva a cabo la migración.

Además de llevar a cabo el funcionamiento de los servicios que son críticos es necesario utilizar una dirección de red IP virtual. Esto permite que los servicios se encuentren siempre disponibles en una misma dirección de red independientemente de la ubicación de los mismos en el clúster, de lo contrario los clientes deberían de conocer todas las direcciones en donde los servicios son migrados cuando ocurre un fallo. Es por ello que cada uno de los nodos debe tener configurada en su interfaz de red la misma dirección IP virtual y además debe ser considerada como un recurso importante a manejar, monitorear y migrar por Heartbeat.

5.4.1. Arquitectura de clúster (Cluster Stack) de Heartbeat

Históricamente cuando se hace referencia a Heartbeat en realidad se está haciendo referencia a un conjunto de herramientas que permiten brindar alta disponibilidad en una arquitectura diseñada en capas. Esta arquitectura se forma principalmente por Heartbeat como capa de comunicación de la arquitectura, un administrador de recursos locales (Local Resource Manager o LRM) y un administrador de recursos de clúster (CRM) en la capa de asignación de recursos, y un agente de recursos (Resource Agents o RA) en la capa de recursos entre otros componentes.

A partir de la versión 2.1.4 de Heartbeat cada uno de estos componentes pasó a ser proyectos independientes en Linux-HA, sin dejar de ser imprescindibles para la construcción de un clúster de alta disponibilidad, siendo la aplicación Heartbeat considerada únicamente como el componente de comunicación entre los nodos. Este tipo de arquitectura modular permite una mayor flexibilidad para la construcción de un clúster dado que puede utilizarse diferentes herramientas que cumplan con la función especificada para cada una de las capas de la arquitectura.

Figura 5.4.1: Heartbeat Cluster Stack.

La arquitectura presentada por Linux-HA es la siguiente:

- **Pacemaker - Módulo CRM:** Este módulo funciona como el pilar fundamental de la estructura de clúster propuesta por Linux-HA. Permite establecer la configuración global del clúster. Se encarga de iniciar, detener, monitorear y configurar los recursos del clúster,

como por ejemplo orden en que debe iniciarse los servicios, la ubicación por defecto de los mismos, establecer una prioridad entre los nodos durante la migración, entre otras cosas.

- **Heartbeat/OpenAIS:** Heartbeat o OpenAIS son dos proyectos que pueden encargarse de la capa de comunicación del clúster. Permiten el intercambio de mensajes entre los integrantes del clúster y la detección de la presencia o ausencia de procesos entre los nodos pares.

- **Agentes de recursos (Resource Agents):** Son un conjunto de scripts para la gestión y acceso a los distintos servicios que pueden ser brindados por el clúster de alta disponibilidad. Estos scripts deben cumplir con ciertas condiciones para poder ser utilizados en la arquitectura de clúster de Linux-HA.

- **Cluster Glue:** Junto con Pacemaker, este elemento es fundamental para la creación de un clúster de alta disponibilidad. Es la capa que permite la interoperabilidad de los distintos módulos que componen la arquitectura propuesta por Linux-HA.

5.4.1.1. Agentes de recursos (Resource Agents):

Los agentes de recursos, tal como se menciona en (Linux-HA, 2011), son interfaces estandarizadas de los recursos que serán administrados por el clúster. Permiten realizar un conjunto de operaciones estándar que se traslada al conjunto de operaciones específicas de un recurso o aplicación, como también permite interpretar sus resultados de éxito o error. Estos scripts son los que utiliza Pacemaker para iniciar, detener o monitorear los recursos del clúster.

Dentro de las operaciones estándar que puede implementar un script de agente de recurso se encuentran:

- *start*: Operación para iniciar el servicio o recurso.

- *stop*: Operación para detener el servicio o recurso.

- *monitor/status*: Esta operación debe verificar el estado del servicio o recurso. Debe indicar únicamente si el recurso está o no funcionando.

Estas operaciones son imprescindibles por Pacemaker ya que las que utiliza principalmente en su funcionamiento: para conocer el estado de un servicio utiliza la operación *monitor/status*, si debe detener servicios relacionados con alguno que haya fallado utiliza la operación *stop*, y cuando los servicios son migrados a otro nodo estos son iniciados en él con la operación *start*.

La implementación de estos scripts es llevada a cabo como scripts Shell de Linux. Son sincrónicos por naturaleza, por ejemplo cuando se inicia un recurso mediante la operación *start* debe esperarse por la totalidad de las operaciones implementadas dentro de ésta para indicar si la operación finalizó con éxito o error. Muchas de estas operaciones (como *start*, *stop* y *monitor/status*) pueden completarse en un tiempo considerable que va de segundos a varios minutos en algunos casos.

Existen tres tipos de agentes de recurso que pueden utilizarse con Pacemaker definidos por el proyecto Linux-HA:

- **Agentes de recurso LSB:** LSB o Linux Standard Base (Linux-HA, 2010) es un proyecto compuesto de varias distribuciones Linux con el objetivo de normalizar la estructura interna de los sistemas operativos Linux siguiendo la especificación POSIX, la especificación Single UNIX y otros estándares abiertos. Los agentes de recurso LSB siguen con estas especificaciones para que puedan ser utilizados bajo cualquier sistema operativo que sigan estas normas. Este estándar define además de las tres operaciones mencionadas un conjunto de operaciones adicionales, los códigos de resultados posibles para cada una, como también los parámetros que estas deben recibir para su funcionamiento.

- **Agentes de recurso OCF:** OCF u Open Cluster Framework (Linux-HA, 2010) es un conjunto de estándares definidos para arquitecturas de Clúster, considerado una extensión Linux Standard Base. Los agentes de recursos OCF cumplen con este tipo estándar. Este tipo de agentes es el más utilizado en las arquitecturas de clúster con Pacemaker. También define su propio conjunto de operaciones, códigos de resultados y parámetros a recibir por cada

una de ellas.

- **Agentes de recurso propios de Heartbeat:** Estos agentes (Linux-HA, 2010) son incluidos por Heartbeat y básicamente son scripts LSB. Poseen únicamente las tres operaciones antes mencionadas (start, stop y monitor).

De esta manera, la capa de recursos (Pacemaker) puede llevar a cabo la administración de cualquier recurso o servicio en el clúster siempre que se cuente con un *Agente de recurso* que cumpla con alguno de los tres tipos de agentes definidos.

5.4.1.2. Cluster Glue

Cluster Glue (Linux-HA, 2010) es un conjunto de librerías, herramientas y utilidades adecuada para la estructura de clúster (Cluster Stack) Heartbeat/Pacemaker. Este componente se encarga de las tareas que no realiza la capa de mensajes, el administrador de recursos de clúster o los agentes de recursos.

Dentro de los componentes que conforman a Cluster Glue se encuentran:

- **Administrador de recursos Locales (LRM):** El administrador de recursos locales es una interface intermedia entre el Administrador de recursos de clúster y los agentes de recursos. No aplica ninguna política, únicamente procesa los comandos recibidos desde el CRM, se los envía a los agentes de recursos y reporta las acciones de éxito o error. En particular el LRM puede:
  - Iniciar un recurso.
  - Detener un recurso.
  - Monitorear un recurso.
  - Reportar el estado de los recursos del clúster.
  - Listar las instancias de todos los recursos con sus estados que se encuentra controlando.

- **STONITH:** Es el mecanismo utilizado para aislar aquellos nodos que son considerados por el clúster como “muertos” o no disponbiles. El significado de STONITH es *Shoot The Other*

Node In The Head, cuya traducción literal es "Dispárale al otro nodo en la cabeza". Este mecanismo se utiliza para evitar el riesgo de que el clúster siga intentando comunicarse con un nodo que se encuentra caído, quitandolo como miembro del clúster.

- **hb_report:** Es una herramienta para el reporte de errores.

- **Cluster plumbing Library:** Es una librería a bajo nivel para comunicaciones intra-cluster.

5.4.1.3. Heartbeat

Heartbeat tiene tres archivos principales de configuración dentro del directorio /etc/ha.d:

- **authkeys:** En este archivo se configuran las claves autorizadas para que los nodos del clúster puedan comunicarse bajo un esquema seguro, evitando por ejemplo la suplantación de identidad. Existen tres esquemas de configuración:

  - **Codificación de los datos mediante CRC.** Es utilizado cuando los nodos se comunican dentro de una red segura, debido a la poca cantidad de recursos de CPU que este tipo de codificación consume.

  - **Codificación de los datos mediante SHA1.** Es utilizado cuando los nodos no están en una red segura y por ende se requiere una mayor protección de la información. La codificación mediante SHA1 requiere de una mayor cantidad de recursos de CPU que la anterior.

  - **Codificación de los datos mediante MD5.** Este método resulta una solución intermedia entre los dos métodos anteriores.

La estructura del archivo es la siguiente:

```
auth<identificador>
<identificador><metodo_autenticacion> [<clave_autenticacion>]
```

Pueden establecerse varios identificadores de seguridad con diferentes métodos de autenticación y especificarse cuál de ellos es el actualmente válido.

- **haresources:** En este archivo se establecen los recursos a los que se dará alta disponibilidad. En él se especifican el host maestro donde se iniciará el servicio, su dirección IP virtual y el nombre del recurso.

La estructura de este archivo es:

```
[Master Host] [IP Virtual] [Recurso]
```

- **ha.cf:** Este archivo es el corazón de la configuración. Se establecen los parámetros principales para el funcionamiento del demonio de comunicación. Entre otras cosas se establece el modo de comunicación entre nodos, se especifican los nodos del clúster, habilitación del CRM y nivel de registro de logs entre otras cosas.

Los parámetros admitidos en este archivo de configuración son:

- **debugfile:** Especifica el directorio y nombre del archivo donde Heartbeat almacena los mensajes cuando funciona en modo depuración.

- **logfile:** Al igual que el parámetro anterior, determina donde guardar los mensajes generados como log.

- **logfacility:** En este parámetro se establece el nivel de detalle del archivo log de Heartbeat.

- **Keepalive:** Es el tiempo (por defecto en segundos, aunque es posible especificarlo en milisegundos escribiendo ms) que transcurre entre la emisión de dos pings hacia cada nodo del clúster para su monitorización.

- **deadtime:** Especifica el tiempo en segundos que deben transcurrir para determinar que un nodo del clúster no está disponible.

- **warntime:** Es el tiempo en segundos que debe transcurrir para indicar el último intento de ping en la monitorización de los nodos. De esta manera se podrá saber que se está próximo a un nodo caído en el clúster.

- **initdead:** Es el tiempo en segundos que puede transcurrir como máximo antes de iniciar Heartbeat en un nodo. Si se excede este tiempo antes de iniciar Heartbeat el nodo es considerado como no disponible.

- **udpport:** Permite especificar el número de puerto para las comunicaciones de tipo unicasty broadcast. Este puerto es el mismo tanto para el envío como recepción de pings de monitorización.

- **bcast:** Interfaz de red que usa Heartbeat para el envío de los pings de monitorización. Este parámetro se utiliza cuando la comunicación del clúster está conformado por más de un par de nodos.

- **ucast:** Su uso es similar al parámetro anterior, con la diferencia de que se utiliza cuando se dispone solamente de comunicaciones entre pares de nodos en el clúster.

- **autofailback**: Este parámetro sirve para especificar el comportamiento de Heartbeat sobre un nodo recuperado de un fallo donde los recursos que funcionaban en él fueron migrados a otro nodo del clúster. Hay dos opciones posibles: que una vez que el nodo que falló se recupera los recursos vuelvan a él, o que permanezcan en el nuevo nodo. Para la primera opción el valor del parámetro es **true** mientras que para la segunda opción el valor del parámetro es **false**.

- **node**: En este parámetro se especifica el hostname de todos los nodos que forman parte del clúster.

- **crm**: Con este parámetro se establece el uso o no del administrador de recursos del clúster (CRM). Los valores admitidos son **yes** o **no**.

Para que Heartbeat funcione correctamente, además de estos archivos de configuración, es necesario editar el archivo /etc/hosts en cada nodo del clúster agregando las direcciones IP y nombres de los nodos que lo componen.

La configuración anterior (*hosts*, authkeys, ha.cf) debe estar replicada en todos los nodos del clúster para su correcto funcionamiento. A continuación se presentará la configuración más importante que determina el comportamiento del clúster que debe realizarse a través de Pacemaker. Esta configuración debe realizarse en un único nodo puesto que luego será replicado al resto de forma automática. La diferencia con los archivos anteriores radica en que éstos dependen en gran medida del nodo en el que nos encontrábamos configurándolos y el que explicaremos a continuación es idéntico para todos.

5.4.1.4. Pacemaker

Como se explicó anteriormente Pacemaker es un gestor de recursos de clúster que permite configurar los recursos, determinar el orden de arranque de los servicios y ubicación por defecto de los mismos, entre otras cosas.

Este gestor es muy importante debido que a través de éste la configuración del clúster se hace más sencilla. En versiones anteriores, dicha configuración debía realizarse editando un archivo XML conocido como cib.xml (Cluster Information Base). En sus últimas versiones esto ya no es así, se puede configurar el clúster mediante el uso del comando *crm* el cual generará finalmente las modificaciones necesarias en el archivo mencionado.

5.4.1.4.1. Estructura del archivo Cluster Information Base

La estructura de este archivo se divide en dos secciones principales:

```
<cib>
      <configuration>
            <crm_config/>
            <nodes/>
            <resources/>
            <constraints/>
      </configuration>
      <status/>
</cib>
```

- **Configuration (Sección estática):** Contiene toda la configuración del clúster, atributos, nodos, recursos, etc.

- **Status (Sección dinámica):** Contiene información actualizada sobre el estado del clúster, es decir, cantidad de nodos online, nodos fallidos, qué servicios han sido migrados, etc.

Dentro de la sección estática encontramos cuatro secciones más:

- **Configuración del CRM o <crm_config>:** Determina el comportamiento del clúster mediante la configuración de distintos atributos y propiedades. Algunas de las propiedades que se utilizan con más frecuencia son:

    - **default-action-timeout:** Tiempo máximo para que un agente realice una operación sobre un recurso. En el caso de que se especifique un timeout particular para una operación en el recurso, este último tiene mayor prioridad sobre el definido por defecto.
    - **symmetric-cluster:** Determina si el clúster es simétrico o asimétrico. En el primer caso, los recursos pueden ser ejecutados en cualquier nodo, en el segundo caso debe especificar en qué nodos pueden ejecutarse qué recursos.

    - **stonith-enabled:** Permite habilitar un dispositivo adicional para terminar o reiniciar un nodo fallido. Esto se realiza para asegurarse la liberación de los recursos que estaban ejecutándose en dicho nodo.

- **stonith-action:** Especifica la acción (apagar o reiniciar) a llevar a cabo por el dispositivo stonith sobre un nodo fallido.

- **cluster-infrastructure:** Especifica la aplicación encargada de la capa de comunicación del clúster, es decir, si se utilizará Hearbeat u Openais.

- **no-quorum-policy:** Especifica la política a seguir en caso de que el clúster pierda el quórum. Se puede especificar, por ejemplo, que el clúster siga funcionando, o que todos los recursos sean detenidos.

- **default-resource-stickiness**: Especifica la adherencia de un recurso a un nodo. Con este valor, se puede especificar la prioridad que tiene un nodo para mantener o recibir un determinado recurso cuando se necesita realizar alguna migración. Puede establecerse el valor INIFINITY para indicar que el recurso nunca debe ser migrado.

- **<nodes> (Nodos):** Determina los nodos pertenecientes al clúster, los nombres introducidos deben estar presentes también en el archivo /etc/hosts. Esta sección permite además agrupar los nodos, limitar los recursos a determinados grupos, establecer ordenamientos, etc.

- **<resources> (Recursos):** Determina los recursos que formarán parte del clúster, y que por ende serán dotados de alta disponibilidad. Al igual que en la sección anterior, los recursos pueden ser agrupados para un manejo más sencillo. Los recursos pertenecientes a un mismo grupo, serán ejecutados en un mismo nodo, y serán iniciados o detenidos en un mismo orden. Es importante destacar que se debe disponer de los agentes de recursos necesarios para poder administrar a los recursos adecuadamente. Sin éstos, será imposible dotar de alta disponibilidad a los recursos definidos en esta sección.

- **<constraints> (Restricciones):** Determina las restricciones o relaciones entre nodos o grupos de nodos y entre recursos o grupo de recursos. A través de éstas, es posible definir de una mejor manera el comportamiento del clúster.

    - **Restricciones de localización:** Determinan para un recurso, en qué nodos pueden ser ejecutados.

    - **Restricciones de colocación:** Determinan qué recursos pueden ser ejecutados en un mismo nodo y cuáles no.

    - **Restricciones de orden:** Determinan el orden en que los recursos son iniciados o detenidos.

5.4.1.4.2. Configuración a través de pacemaker

Para configurar el archivo cib.xml a través de la aplicación *crm* de Pacemaker, primero es necesario que Heartbeat este corriendo en cada uno de los nodos del clúster. Una vez realizado esto, podemos acceder a las siguientes opciones del *crm*:

- **Cib:** Permite editar, eliminar, actualizar el archivo cib en el resto de los nodos del clúster. También permite separar la configuración en distintos archivos y aplicarlos cuando se crea necesario.

- **Resource:** Permite conocer el estatus de cada recurso en el clúster, modificarlo, y migrarlo entre otras cosas.

- **Node:** Permite conocer el estado de cada nodo en el clúster, modificarlo, eliminar su configuración, etc.

- **Configure:** Permite administrar los nodos, los recursos, los grupos y las restricciones del clúster.

- **Ra:** Permite listar información relativa sobre los agentes de recursos. Los agentes de recursos permiten realizar las acciones de start, stop, status y monitor sobre los recursos.

- **Status:** Permite conocer el estado del clúster, saber que nodos se encuentran online, qué recursos poseen, cuáles son los últimos fallos detectados, etc.

5.4.1.4.3. Posibles configuraciones de redundancia de Clúster

Heartbeat+Pacemaker no hace suposiciones acerca del entorno en el cual va a trabajar, lo que le permite soportar prácticamente cualquier tipo de configuración de redundancia incluyendo configuraciones Activo/Activo, Activo/Pasivo, N+1, N+M, N:1 y N:N.

5.5. Linux Virtual Server - LVS

Linux Virutal Server (o LVS) es un proyecto que surge bajo la necesidad de desarrollar un servidor de alto rendimiento capaz de proporcionar estabilidad y robustez a sistemas implementados bajo arquitecturas de clustering en sistemas operativos Linux. Trabaja a nivel de **Kernel** y brinda ventajas tales como poder incluir nuevos servidores al cluster sin importar el sistema operativo que éstos utilicen, siempre y cuando tengan soporte bajo el protocolo TCP/IP, ni tampoco la disposición geográfica de los mismos, ya sea que los servidores estén interconectados dentro de una misma LAN o dispersos en diferentes redes WAN. La escalabilidad de los sistemas se logra en forma transparente agregando o quitando servidores al cluster, y la alta disponibilidad es provista mediante la detección de la falla de un servidor o un servicio y su adecuada reconfiguración. LVS desarrolla un sistema avanzado de balanceo de carga por software a nivel IP (IPVS), balanceo de carga por software a nivel de aplicación y componentes para la gestión de clusters (KTCPVS).

IPVS es el sistema avanzado IP de balanceo de carga por software que está incluido en el propio Kernel de Linux en las versiondes 2.4 y 2.6. Es implementado en un equipo por delante del clúster que contiene los servidores reales que brindan un conjunto de servicios. Tiene la capacidad de dirigir las peticiones de los servicios basados en TCP/UDP hacia los servidores reales creando en apariencia que los servicios trabajan en un único servidor, denominado servidor virtual, en una única dirección IP. La aplicación encargada de administrar y de crear las directivas en el Kernel de Linux es IPVSadm. El equipo encargado de realizar el balanceo de carga es denominado Linux

Director.

KTCPVS es un proyecto que se encuentra aún en desarrollo que también es incluido en el propio Kernel, se encarga del balanceo de carga de software a nivel de aplicación.

5.5.1. Implementación del servidor virtual

El servidor virtual puede implementarse de tres formas de acuerdo a tres técnicas diferentes de balanceo de carga:

- A través de NAT
- A través de encapsulamiento IP
- A través de enrutamiento directo

5.5.1.1. Servidor Virtual a través de NAT

La ventaja de servidores virtuales a través de NAT es que los servidores reales pueden operar bajo cualquier sistema operativo que trabaje con el protocolo TCP/IP, los servidores reales pueden utilizar una dirección privada de red.

La desventaja recae sobre la escalabilidad, ya que esta se encuentra limitada. El balanceador de carga puede generar un cuello de botella cuando el número de servidores se encuentra cerca de los 20 servidores o más, dado que los paquetes de peticiones y los paquetes de respuesta deben ser traducidos (NAT) por el balanceador.

Suponiendo que la longitud promedio de un paquete TCP es de 536 Bytes, el retraso promedio por la traducción de los paquetes ronda los 60us, el throughput máximo del balanceador de carga es de 8.93 Mb/s. Asumiendo que el throughput promedio de un servidor real es de 400Kbytes/s, el balanceador de carga puede administrar 22 servidores reales.

El servidor virtual puede alcanzar el cumplimiento de varias peticiones de varios servidores. Aun cuando este se transforme en un cuello de botella en el sistema. Este problema es resuelto con las otras técnicas de balanceo de carga posibles.

5.5.1.2. Servidor Virtual a través de Encapsulamiento IP

En la técnica anterior todas las peticiones y las respuestas necesitan ser procesadas en el balanceador, generándose un posible problema de cuello de botella cuando el número de

servidores reales asciende a 20 o más. Cualquier servicio brindado a través de Internet tiene la particularidad que el tamaño de los paquetes de peticiones es mucho menor al tamaño de los paquetes de sus respectivas respuestas dado que transportan gran cantidad de información.

En esta técnica, el balanceador de carga solamente se encarga de programar las peticiones a los distintos servidores reales, y estos son los que envían las respuestas directamente a los clientes. De esta forma, el balanceador puede manejar un número de peticiones mucho más elevado, un valor aproximadamente mayor a 100 servidores, sin que pueda generarse un cuello de botella en el sistema.

El throughput máximo que este tipo de servidor virtual puede alcanzar ronda a 1GBps, aun cuando el dispositivo de red que éste posea sea de 100Mbps full dúplex.

Las características de encapsulamiento IP pueden utilizarse para construir un servidor virtual de alto rendimiento. Sin embargo, todos los servidores reales deben tener el protocolo de encapsulamiento IP habilitado, el cual puede generar un procesamiento extra (aunque generalmente es mínimo). El encapsulamiento permite que los servidores reales no necesariamente deban encontrarse en el mismo lugar geográfico que el balanceador de carga.

5.5.1.3. Servidor virtual a través de enrutamiento directo

Al igual que las características de encapsulamiento, el Linux Director procesa solamente una parte de la conexión entre los clientes y los servidores reales a través de enrutamiento directo. Comparado con la técnica anterior, esta técnica no tiene un procesamiento adicional por el encapsulamiento, pero requiere que uno de los dispositivos de red del balanceador deba estar conectado en el mismo segmento físico en el que están conectados los servidores reales. En la siguiente tabla puede verse en forma resumida las ventajas y desventajas que presenta cada una de las técnicas recién mencionadas:

	A través de NAT	A través de Encapsulamiento IP	A través de enrutamiento directo
Tipo de red	Privada	LAN/WAN	LAN
Número de servidores	Bajo, 10-20	Alto	Alto
Gateway	Balanceador de carga	Router	Router

Con el propósito de realizar un escenario más completo y complejo, para este trabajo se seleccionó desarrollar la configuración de un servidor virtual a través de encapsulamiento IP. En la siguiente sección se hará una explicación más completa sobre este tipo de técnica de balanceo de carga.

5.5.2. Servidor virtual a través de encapsulamiento IP

El encapsulamiento IP se basa en encapsular paquetes IP dentro de otro paquete IP, lo que permite que paquetes destinados hacia una dirección en particular puedan ser contenidos dentro de un nuevo paquete como si estos fuesen su payload y enviarlos hacia otra dirección. La información relacionada a este tipo de protocolo de encapsulamiento se encuentra en el RFC 2003 (Perkins, 1996).

Una de las características principales en este tipo de balanceo es que las peticiones son entregadas al balanceador de carga, el cual se encarga de redirigirlo hacia un servidor real y éste finalmente envía la respuesta final.

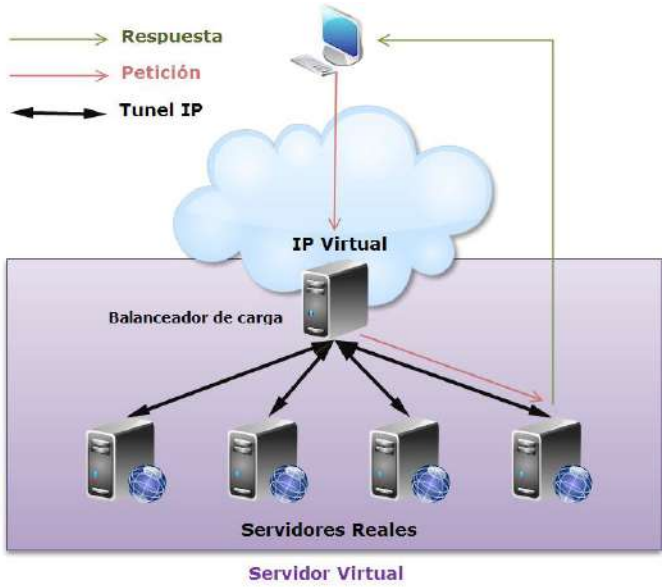


Figura 5.5.1: Tratamiento de peticiones.

Todas las peticiones hacia el servidor virtual son dirigidas a una dirección IP virtual. El balanceador de carga se encarga de examinar la dirección destino y puerto de las peticiones. Si coincide con alguno de los servicios que brinda el servidor virtual, el balanceador selecciona a uno de los

servidores reales de acuerdo a un algoritmo de planificación configurado. Por otra parte, la conexión se agrega en una tabla hash encargada de mantener información de las conexiones concurrentes. De esta forma, el balanceador de carga puede saber cuántas y qué conexiones están siendo atendidas por cada servidor, lo que le permite distribuir las peticiones de una forma eficiente.

Una vez que se selecciona un servidor real para atender una petición, el balanceador la encapsula y se la envía. Cuando arriba otro paquete de la misma conexión y el servidor real se encuentra en la tabla hash, el paquete es enviado al mismo servidor. De esta manera todos los paquetes pertenecientes a una misma conexión son siempre atendidos por el mismo servidor real. El servidor real desencapsula la petición, la procesa y envía una respuesta directamente al usuario final a través de una tabla de ruteo interna. Una vez que la conexión finaliza o el tiempo de expiración de la misma llega a su fin, ésta es eliminada de la tabla hash del balanceador.

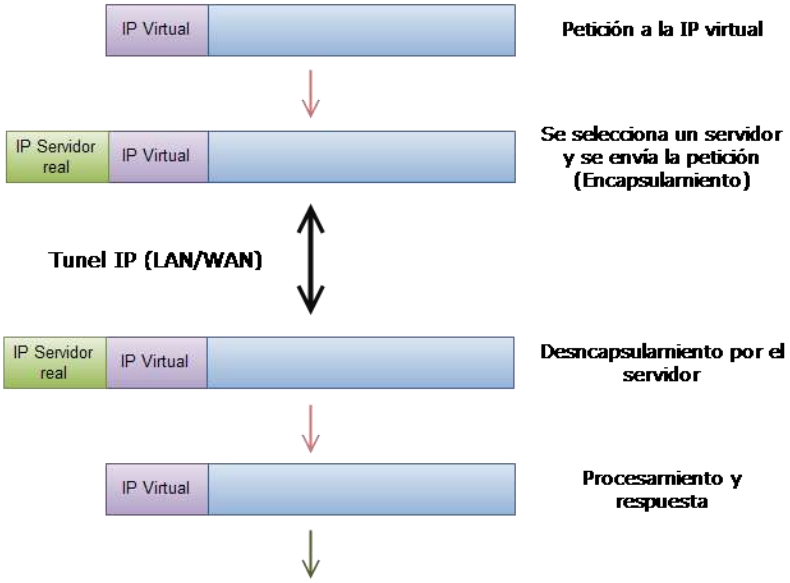


Figura 5.5.2: Encapsulamiento y desencapsulamiento IP.

Los servidores reales pueden tener cualquier dirección de red, en cualquier disposición geográfica, pero deben soportar el protocolo de encapsulamiento IP. Sus dispositivos de túnel son configurados de forma que puedan recibir los paquetes encapsulados apropiadamente, y la dirección virtual debe ser configurada en un dispositivo que no utilice ARP debido a que los

servidores no se encuentran obligatoriamente en el mismo segmento de red. De contar con ARP los nodos que estén en distintos segmentos no serían encontrados.

5.5.3. Ldirectord

Ldirectord (Horms Solutions, 2010) es un demonio para monitorear y administrar los servidores reales en un clúster LVS de balanceo de carga. Suele ser utilizado como un recurso para Linux-HA, pero también puede ser ejecutado desde la línea de comandos. El archivo de configuración de Ldirectord específica las características del servicio que debe proporcionar y los servidores a los que debe monitorear, además de leer las directivas establecidas en el Kernel a través de IPVSadm.

La monitorización se realiza controlando la vida de los servidores mediante una petición a una URL conocida y verificando que la respuesta obtenida es la esperada. Esta petición se realiza de forma periódica, cuando se detecta que un servidor ha fallado, se lo remueve del clúster hasta que este se recupere de su falla.

5.5.3.1. Archivo de configuración

El archivo de configuración del balanceador de carga se encuentra en el directorio /etc/ha.d/ldirectord.cf y posee las siguientes opciones de configuración:

- **Checktimeout:** Define el tiempo de espera máximo para una verificación. Si el timeout es excedido sin obtener la respuesta esperada, el servidor monitoreado se declarará como muerto.

- **Checkinterval:** Define el tiempo en segundos entre una verificación y otra.

- **Checkcount:** Define el número de veces que el control debe ser realizado antes de considerar que el servidor ha fallado. Esta opción solo funciona con los controles de ping.

- **Autoreload:** Define si Ldirectord debe controlar si ha habido cambios en el archivo de configuración. Si esto ocurre, la configuración será cargada automáticamente sin necesidad de tener que reiniciar al Ldirectod.

- **Fallback:** Define el servidor con un webservice al cual se debe redirigir si todos los servidores se encuentran caídos. Por lo general, este será el localhost.

- **Logfile:** Define el archivo y el directorio donde se almacenaran los logs de ldirectord.

- **Supervised:** Define que el demonio no se ejecute en background. Esta opción es útil para poder observar los mensajes del log en tiempo real.

- **Quiescent:** Define si cuando se detecta un servidor caído debe ser removido del LVS o si se debe cambiar su peso a cero. Si se cambia el peso de un servidor a cero, éste no recibirá nuevas conexiones, a menos que tenga conexiones persistentes, en dicho caso solo conexiones de clientes existentes seguirán siendo redirigidas a dicho servidor hasta que el tiempo de expiración de persistencia expiré. Si se remueve del LVS, ninguna conexión será redirigida a dicho servidor hasta que éste se recupere.

- **Virtual:** Define un servicio virtual. Puede definirse indicando la dirección IP (o hostname) y el puerto (o nombre del servicio) o la marca del firewall. La marca del firewall es un número entero mayor a cero con la que se han marcado a los paquetes que deben ser monitoreados. Esta directiva puede considerarse como una sección que contendrá a su vez más directivas que afectarán al servicio virtual.

Las directivas que pueden ser incluidas dentro de Virtual son las siguientes:

- **Checktype:** Define el tipo de monitoreo a realizar, pueden ser negotiate, connect, N, ping, off, on.

- **Negotiate:** Consta de enviar una petición y comprobar un texto recibido.

- **Connect:** Consta únicamente de realizar una conexión TCP/IP.

- **N:** Es un número y suele combinarse con el tipo Connect. Indica el número máximo de

intentos de conexión que podrán realizarse para determinar si el servidor está vivo.

- **Ping:** Se utilizará el mensaje ICMP Ping para comprobar la disponibilidad de un servidor.
- **Off:** Significa que no se llevará a cabo ningún control y ningún servidor será activado.
- **On:** Significa que no se llevará acabo ningún control y los servidores serán siempre activados.
- **Service:** Tipo de servicio a monitorear cuando el checktype está establecido en negotiate. Los valores posibles son: ftp, smtp, http, pop, nntp, imap, ldap, https, dns, mysql, pgsql, sip, none.
- **Checkport:** Número de puerto a monitorear.
- **Request:** Dirección URI a monitorear. Se realizará una petición a dicha URI cada checkinterval para cada servidor real.
- **Receive:** Determina una expresión regular que validará la respuesta de cada servidor real para determinar si se encuentra vivo o muerto.
- **Httpmethod:** Determina el método HTTP que deberá ser usado. Puede ser GET o HEAD.
- **Virtualhost:** Determina el atributo Host del encabezado HTTP en la petición de monitoreo.
- **Login:** Determina el nombre de usuario para conectarse a servidores FTP, POP, IMAP, MySQL, PostgresSQL, etc.
- **Passwd:** Determina la contraseña a utilizar para lograr conectarse junto con el nombre de usuario a un servidor.

- **Database:** Determina el nombre de la base de datos sobre la cual se realizará una consulta.

- **Scheduler:** Nombre del esquema a utilizar por LVS para el balanceo de carga. Por defecto es WRR (Weighted Round Robin).

- **Persistent:** Determina los segundos para conexiones de cliente persistentes.

- **Netmask:** Determina el número de máscara de red para lograr una mayor granularidad de conexiones de clientes persistentes.

- **Protocol:** Determina el protocolo a utilizar. Puede ser TCP, UDP o FWM (Firewall mark).

5.5.4. IPVSadm

Como se mencionó, IPVS actúa como balanceador de carga a nivel de la capa de transporte dentro del Kernel de Linux y se encarga de redirigir las peticiones hacia los servidores reales implementando algún algoritmo de planificación de balanceo de carga.

IPVSadmin (The Linux Virtual Server Project, 1998) es la aplicación en Linux para administrar este módulo del Kernel mediante el cual se realiza la administración de IPVS. Este comando es utilizado tanto en el balanceador de carga como en los servidores reales. La ejecución de la aplicación se realiza con la siguiente sintaxis:

- Balanceador de carga:

```
ipvsadm COMMAND [protocol] service-address[scheduling-method] [persistence
options]
```

- Servidor Real:

```
ipvsadm command [protocol] service-addressserver-address [packet-forwarding-
method][weight options]
```

Entre los comandos más importantes se encuentran:

- A: Permite agregar un servicio virtual.
- C: Limpia la tabla deLVS. Es importante ejecutar el comando ipvsadm -C antes de iniciar una nueva configuración.
- a: Permite agregar un servidor real a un servicio virtual.
- d: Permite eliminar un servidor real a un servicio virtual.

En cuanto a los parámetros aceptados se encuentran:

- -t *service-address*: Se utiliza para indicar que se agregará un servicio TCP. *service-address* debe ser especificada de la forma host[:port], siendo host una dirección IP o un hostname y port el número de puerto o el nombre del servicio (por ejemplo http).
- -u *service-address*: Se utiliza para indicar que se agregará un servicio UDP.
- -s *scheduling-method*: Se utiliza para especificar el algoritmo de planificación a utilizar. Los valores válidos para *scheduling-method* son:
    - rr (Round-Robin)
    - wrr (Weighted Round-Robin)
    - lc (Least-Connection)
    - wlc (Weighted Least-Connection)
    - lblc (Locality-Based Least-Connection)
    - lblcr (Locality-Based Least-Connection with Replication)
    - dh (Destination Hashing)
    - sh (Source Hashing)
    - sed (Shortest Expected Delay)
    - nq (Never Queue)
- -r *server-address*: Se utiliza para especificar la dirección IP de un servidor real, y opcionalmente puede agregarse un número de puerto.
- -w *weight*: Se utiliza para especificar el peso de un servidor real. El peso es un valor numérico que indica la capacidad de procesamiento del servidor, relativa al conjunto de servidores. El rango de valores válidos de *weight* es [0, 65536]. El valor por omisión es 1.

Un servidor con peso 5 indica que tiene mayor capacidad de procesamiento que un servidor con peso 2, y por tanto recibirá mayor número de peticiones. El balanceo de peticiones de forma equitativa, es decir la misma cantidad de conexiones por servidor, se puede realizar colocándole a todos ellos el mismo valor de peso.

5.6. Monitoreo

El monitoreo es una actividad muy importante dentro del concepto de alta disponibilidad. Si bien el monitoreo no es expuesto explícitamente cuando se hace referencia a los esquemas de replicación, balanceo o restauración aplicables para brindar alta disponibilidad, se conoce que debe realizarse para determinar por ejemplo si un servicio está o no funcionando o si un nodo dejó de responder dentro del clúster.

La mayoría de las soluciones software cuentan con un componente destinado a realizar esta operación. Por ejemplo, en el Cluster Stack de Heartbeat esta operación es realizada en conjunto por el Administrador de recursos de Clúster (CRM), el Administrador de Recursos Locales (LRM) y los Agentes de Recursos, que envían la orden y el resultado del monitoreo a través del componente encargado de la capa de comunicación como lo es Heartbeat. El CRM necesita conocer los estados de los servicios y de los nodos del clúster para determinar si los servicios fueron interrumpidos y donde pueden ser migrados en caso de fallo. Para ello envía la orden de monitoreo al LRM de los nodos registrados en el clúster. Cada LRM ejecuta la orden de monitoreo a través de los Agentes de Recursos, encargados de conocer cómo realizar esta operación para cada servicio en particular, y retorna la respuesta correspondiente. Toda la comunicación entre los componentes se realiza a través de Heartbeat.

En cuanto a lo que se refiere al monitoreo de servidores Asterisk, debería contarse con una herramienta de monitoreo capaz de verificar todos los aspectos fundamentales para determinar si el servicio de telefonía IP está disponible. Estos aspectos abarcan las capacidades de registración, ubicación de usuarios y realización de llamadas visto del lado del cliente, sumado a esto debe tenerse en cuenta la configuración del clúster Asterisk abarcado por la comunicación entre servidores Asterisk, comunicación a una base de datos compartida y otros servicios configurados. Como Asterisk no cuenta con una herramienta propia de monitoreo ni con un script de arranque

que sea válido para Agentes de Recursos LSB, OCF o propios de Hearbeat se ha visto la necesidad de crear un script propio que cuente con esta capacidad necesaria de monitoreo. Para ello se utilizó la herramienta SIPp creada específicamente para realizar pruebas sobre servidores Asterisk, con la capacidad de crear escenarios que simulan tanto el funcionamiento de un cliente de telefonía IP como también el funcionamiento de un servidor Asterisk.

En el 0 se explicará de que se trata esta herramienta y como fue utilizada para crear un escenario de monitoreo que da paso a la creación del script que será utilizado por los Agentes de Recursos y de esta manera contar con los recursos necesarios para montar el clúster de servidores Asterisk.

5.7. Clúster Asterisk

La principal razón para crear un clúster en una central telefónica radica en la posibilidad de escalar el número de llamadas simultáneas que puede gestionar una central telefónica agregando simplemente un servidor adicional. La segunda razón, e igual de importante, es la posibilidad de seguir gestionando llamadas por el resto del clúster ante la caída de uno de los servidores sin que este hecho afecte significativamente el funcionamiento de la central telefónica. Esto requiere además que exista un balanceo de carga entre los servidores para que la tarea de gestionar las llamadas esté repartida dentro del clúster y no recaiga siempre sobre un mismo servidor.

Durante el diseño y construcción de una central telefónica deben tenerse en cuenta varios factores como el tipo de prestación que se dará, la cantidad máxima de llamadas que pueden ser atendidas en un mismo instante, la cantidad de clientes estables, entre otros. Debido a estos factores, los casos más comunes de diseño planteados por Leif Madsen (Madsen, 2008) pueden resumirse en:

- Servidor simple
- Multiservidor
- Clúster de servidores

A continuación se expondrán cada uno de ellos con sus ventajas y desventajas en su construcción.

5.7.1. Servidor simple

Un servidor simple es la arquitectura más simple de construir. Este se encarga de toda las tareas. Como es lógico, es mucho más sencillo de configurar pero no es escalable y al poseer un único punto de falla, si este servidor deja de funcionar todas las comunicaciones también lo hacen. Además, el número de llamadas concurrentes que pueden ser gestionadas dependen de la capacidad de procesamiento del servidor.

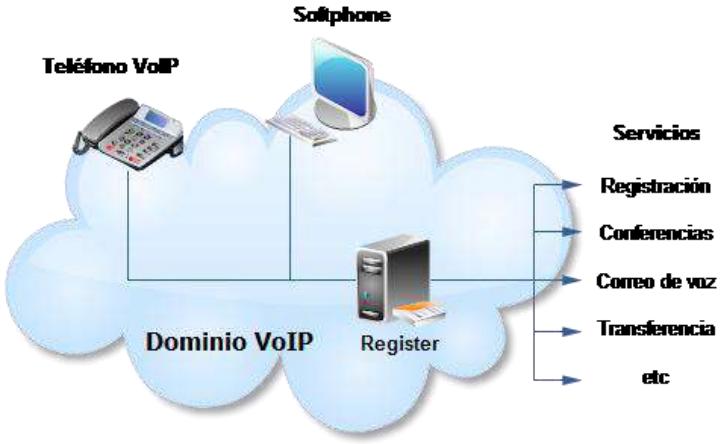


Figura 5.7.1: Arquitectura de Servidor simple.

5.7.2. Multiservidor

La arquitectura multiservidor permite mejorar un poco la escalabilidad de la central telefónica. Se compone de varios servidores configurados para manejar uno o más servicios. Un nodo central se encarga de las registraciones y de redirigir las peticiones de servicio a los servidores correspondientes. Mejora la cantidad de llamadas concurrentes que pueden gestionarse, pero sigue existiendo un único punto de falla del que depende toda la central telefónica para seguir funcionando. De fallar alguno de los servidores de servicio, solo se estaría denegando el servicio que éste brinda.

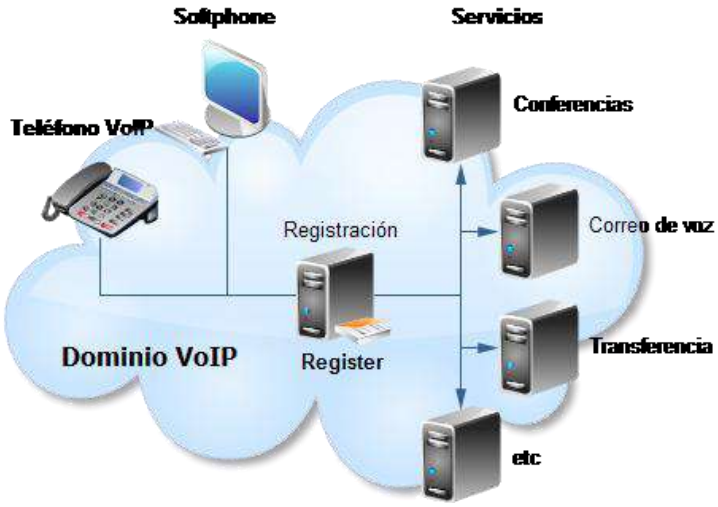


Figura 5.7.2: Arquitectura de Multiservidor.

5.7.3. Clúster de servidores

El clúster de servidores es la arquitectura más compleja para construir una central telefónica. El clúster puede ser construido con las siguientes características:

- Centralizado
- Distribuido
- Híbrido

5.7.3.1. Clúster de servidores Centralizado

Un clúster centralizado está compuesto por un nodo central de registración el cual distribuye las llamadas al resto de los servidores de acuerdo a la carga actual de cada uno de ellos. Todas las registraciones se llevan a cabo sobre un único lugar, el nodo central, por lo que no existe un procesamiento extra para determinar la ubicación de un extremo cuando se realiza la llamada.

Como puede verse, el hecho de ser centralizado brinda la ventaja de poder verificar la carga de los servidores antes de asignar la llamada a un servidor. La central parece ser más sencilla de administrar. Sin embargo, posee todas las desventajas inherentes a un sistema centralizado. Además, existe un umbral máximo de registraciones que el nodo central puede manejar.

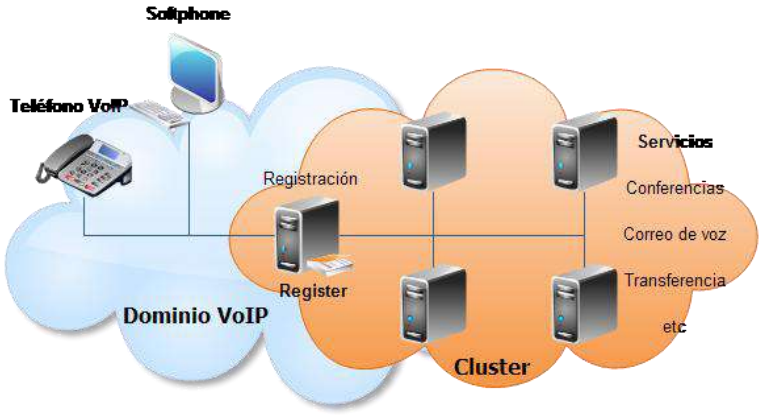


Figura 5.7.3: Clúster de servidores Centralizado.

5.7.3.2. Clúster de servidores distribuido

En un clúster distribuido las llamadas pueden registrarse en cualquiera de los servidores del clúster. Si un extremo falla en la registración en un servidor, puede intentarlo con algún otro del clúster. De esta manera se crea un ambiente tolerante a fallas y no hay denegación de servicios.

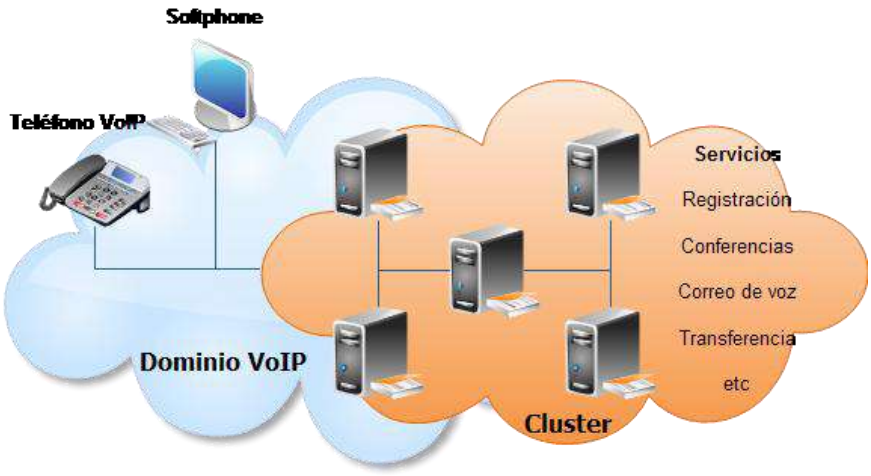


Figura 5.7.4: Clúster de servidores Distribuido.

Cuando se realiza una llamada, Asterisk debe encontrar los puntos de registración de ambos extremos para poder ponerlos en contacto. Esto es posible utilizando DUNDi.

Como desventaja en este tipo de arquitectura es muy difícil controlar el nivel de carga que posee cada servidor. Requiere de una planificación muy cuidadosa para mantener escalabilidad y por

otro lado las arquitecturas tolerantes a fallas tienen limitaciones de escalabilidad.

5.7.3.3. Clúster de servidores Híbrido

Un clúster híbrido combina las ventajas de las dos arquitecturas anteriores. Mantiene separado el proceso de administración de llamadas del proceso de registración.

El proceso de registración puede ser atendido o planificado de dos manteras diferentes:

- Todos los servidores son capaces de realizar la registración, pero sólo uno es el principal. Si este cae, cualquiera del resto de los servidores toma la posta y se transforma en el nuevo servidor de registración en el clúster. Como desventaja podemos observar que todos los servidores en el clúster deben estar al tanto de las registraciones, para que en el caso de caer el servidor de registración actual no se pierdan las registraciones realizadas.

- La otra manera es tener un par de servidores separados del clúster especialmente encargados de la registración. Este par de servidores trabajarán como Failovers. En el caso de que el servidor primario caiga, el otro empezará a responder a las registraciones hasta que el servidor primario se recupere.

La arquitectura híbrida combina lo mejor de las arquitecturas centralizada y distribuida y brinda un mejor sistema tolerante a fallas. La escalabilidad es mucho mayor y mejor, bastaría con agregar un servidor al clúster encargado del procesamiento de llamadas o agregar un nuevo servidor al conjunto de servidores de registración.

La desventaja claramente visible es la topología que resulta de combinar ambas arquitecturas, siendo complicada de configurar y de mantener. Además, se requiere de más tecnología para llevar a cabo ciertas funciones, como Heartbeat para conocer el estado de los servidores de registración y un balanceador de carga para administrar la carga del clúster, y esto puede convertirse en varios puntos de falla adicionales.

Decidimos utilizar una arquitectura híbrida para nuestro trabajo final, siendo la más interesante y completa en aspectos técnicos, además de permitirnos emplear conceptos adquiridos en distintas asignaturas cursadas durante la carrera de Ingeniería en Sistemas en la UNLPam.

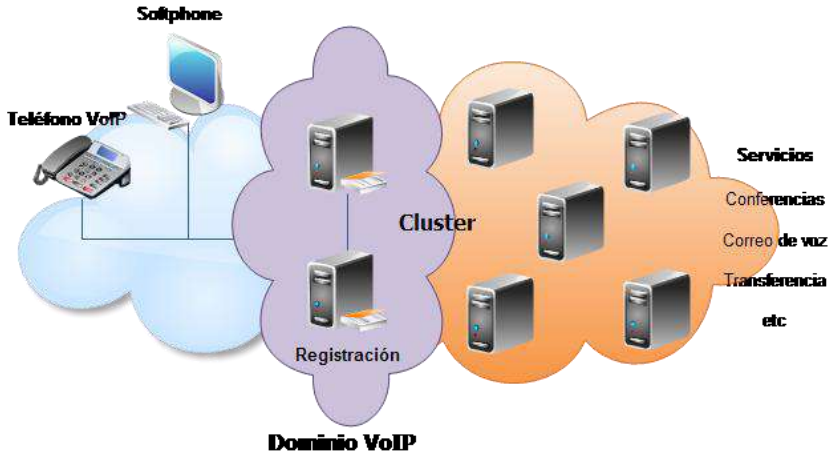


Figura 5.7.5: Clúster de servidores Híbrido.

Las herramientas que pueden utilizarse en VoIP para ambientes distribuidos son:

- DUNDi
- Realtime Asterisk Architecture
- func_obdc
- Voicemail centralizado: IMAP o ODBC

Debido que las herramientas DUNDi y Realtime fueron las seleccionadas para implementar la arquitectura de clúster de servidores híbrido, a continuación se hará mención y explicación de su funcionamiento.

5.8. DUNDi (Distributed Universal Number Discovery)

DUNDi (Distributed Universal Number Discovery), como se define en (Spencer, 2004), es un protocolo diseñado para resolver números telefónicos en recursos distribuidos en Internet para contactarlos a través de sistemas peer-to-peer confiables. Además, a diferencia de sistemas centralizados, DUNDi funciona sin la necesidad de un nodo de autoridad centralizado y por lo tanto sin un único punto de falla ni una arquitectura compleja.

Con este protocolo, los sistemas comparten un directorio "telefónico" que es construido a través de una organización o la red entera, sin perder la confiabilidad en la precisión de las rutas

suministradas ni la seguridad de las consultas y respuesta en el grupo de confianza generado.

Si bien DUNDi puede ser utilizado en otras aplicaciones para resolver rutas para otro tipo de información, como direcciones de correo electrónico, las implementaciones deben desarrollar optimizaciones bajo la presunción de que la consulta es, de hecho, sobre un número telefónico y debe confiar en las heurísticas del protocolo.

Este protocolo es valioso en redes donde la pérdida de un nodo es significativa. Además, es muy práctico para construir relaciones confiables entre organizaciones independientes que desean comunicarse entre sí pero la centralización o la delegación de la información requerida no es práctica o es imposible.

Lo más importante aún, DUNDi es valioso en redes donde un número telefónico puede trasladarse de un proveedor a otro, como es el caso de la portabilidad numérica que actualmente se está desarrollando en nuestro país.

5.8.1. Funcionamiento

DUNDi provee elementos de información y mensajes diseñados para permitir flexibilidad a las aplicaciones que lo implementen en el desarrollo de formas que permitan minimizar el volumen de tráfico y maximizar el número de resultados que pueden ser obtenidos a través de la aplicación. No es utilizado para el establecimiento de la llamada ni para el control de voz, únicamente para ubicar el recurso al que debe trasladarse una llamada ya que para el control y el establecimiento existen protocolos específicos como SIP, IAX/IAX2 o H.323.

Cada nodo dentro de este protocolo es identificado a través de un identificador único y global (EID – Entity Identifier) de seis bytes, cuyo valor generalmente es la dirección MAC de la interface que se encuentre conectada al sistema.

DUNDi está diseñado para proveer seguridad incorporando criptografía fuerte, mediante RSA y AES para la encriptación y RSA para la autentificación de información, que asegura la privacidad y la seguridad de los mensajes transmitidos entre los nodos. Además, brinda contextos explícitos que permiten la separación o superposición de directorios y permitir que ciertos contextos sean compartidos con sólo algunos nodos o grupos de nodos. Sin embargo, DUNDi no prevé la inclusión

de información falsa o no válida proveniente de otros nodos que pertenecen al grupo confiable en la red.

Este protocolo utiliza un formato de paquete simple con elementos de información que proveen atributos adicionales necesarios para los mensajes que el protocolo envía. Trabaja sobre el protocolo de transporte UDP en el puerto "bien conocido" 4520.

La construcción de un grupo DUNDi confiable se basa en un conjunto de nodos donde cada uno de ellos está conectado al menos a uno de los nodos pertenecientes al grupo. En la práctica, cada nodo debería estar conectado por lo menos a dos nodos del grupo para lograr brindar redundancia.

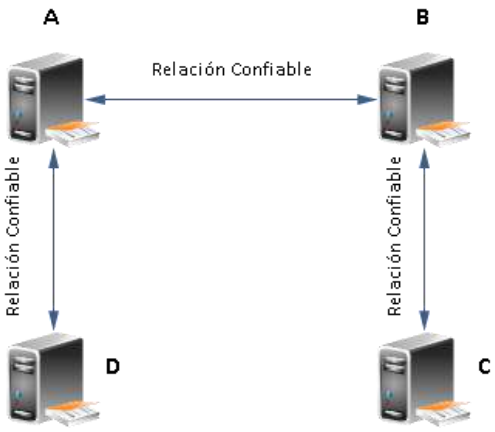


Figura 5.8.1: Esquema de grupo DUNDi confiable.

En un esquema como en la Figura 5.8.1, existen varias relaciones confiables donde sobresalen las siguientes:

1. La relación confiable compuesta entre el nodo A y los nodos B y D.
2. La relación confiable compuesta entre el nodo B y los nodos A y C.

Ante la búsqueda de un número telefónico en este esquema por parte del nodo B, este primero verifica que el número no se encuentre dentro de su tabla de enrutamiento interna o caché. De no encontrarlo, este nodo construye y envía una consulta hacia los nodos A y C con los cuales posee una relación confiable. Estos nodos realizan la misma operatoria verificando que el número se encuentre en sus tablas de enrutamiento locales o sus caché. Si alguno de ellos posee la ruta para

llegar al número solicitado responde a la consulta con la ruta. Si fuese el caso de que el nodo A y el nodo C no poseen la información requerida, estos pueden trasladar la consulta a sus relaciones confiables, en este caso el nodo A enviaría la consulta al nodo D y esperaría por la respuesta para enviársela al nodo B. Puede darse el caso en donde más de un nodo puede proporcionar la respuesta a la consulta incluso si estas poseen el mismo peso.

Es por esta razón que las implementaciones del protocolo deben intentar minimizar el número de consultas que se realizan en un sistema confiable sin perjudicar la precisión de las respuestas obtenidas. El número de consultas puede ser reducido utilizando cacheo de respuestas o bien realizando una "poda" en el árbol de relaciones confiables a través del diseño de una buena arquitectura.

5.8.2. Transacciones

Una transacción en DUNDi es un dialogo completo entre dos nodos y es identificada por ambos extremos a través un identificador único de 16 bit arbitrario con valor entre 1 y 65535, que debe ser aleatorio para aumentar la seguridad. Este identificador es conocido como número de transacción de origen/destino para que cada extremo, origen/destino, pueda ser identificado de manera aislada. El valor 0 como número de transacción destino se reserva para el mensaje de inicio de dialogo, debido a que ese valor aún no es establecido hasta que el extremo destino del mensaje responde con dicho valor.

Una transacción debe comenzar por la transmisión de un comando y no por la transmisión de una respuesta. La transacción puede finalizar en cualquier momento cuando uno de los nodos participantes del dialogo transmite un mensaje de finalización, además de un ACK, con el bit **F** o Final establecido en el mensaje. La recepción de este mensaje debe ser correspondido con un mensaje ACK. Los nodos no deben continuar enviando mensajes luego de haber recibido el mensaje con el bit **F** establecido, con excepción del mensaje ACK para dicho mensaje donde debe obligatoriamente contener también el bit **F** establecido.

Cada mensaje incluye además dos valores de 8 bits de números de secuencia de entrada y salida. Cada nodo tiene un número de secuencia de salida independiente que es iniciado con el valor 0 para el primer mensaje que envía en la transacción. Este valor se incrementa cada vez que se envía

un mensaje, excepto cuando se envía un mensaje ACK. El número de secuencia de entrada representa el próximo valor que debería ser enviado en el siguiente mensaje como número de secuencia de salida por parte del otro extremo del dialogo.

Tras la recepción de un mensaje que no es un ACK o mensaje INVÁLIDO con el número de secuencia de salida igual al valor de número de secuencia de entrada esperado, el receptor debe incrementar el valor del número de secuencia de entrada esperado y enviar un mensaje con el valor del número de secuencia de entrada actualizado. Si no hay mensajes que enviar tras la recepción de un mensaje, se debe enviar un mensaje ACK.

Por otro lado, tras la recepción de un mensaje que no es un ACK o mensaje INVÁLIDO con el número de secuencia de salida igual al valor de número de secuencia de entrada esperado, el receptor debe responder inmediatamente con un comando ACK, con el número de secuencia de salida establecido con el valor del número de secuencia de entrada del mensaje recibido y el número de secuencia de entrada establecido con el valor del número de secuencia de entrada esperado sin ser incrementado.

Todo mensaje recibido donde el número de secuencia de salida no concuerda con el número de secuencia de entrada esperado debe ser ignorado.

DUNDi posee una ventana de tiempo de transmisión de valor 1, lo que significa que no puede enviarse otro mensaje hasta que sea recibido un mensaje por parte del otro participante conteniendo el número de secuencia de entrada establecido que es el número de secuencia de salida que deberá contener el próximo mensaje que le sea enviado.

Un requerimiento DUNDi puede ser retransmitido hasta 10 veces en intervalos de tiempo determinados por quien implementa el protocolo, que no debe ser mayor a un segundo. Si un requerimiento no es respondido luego de 10 retransmisiones o 10 segundos, la transacción debe ser considerada como cerrada sin retransmitir ni esperar más mensajes.

5.8.3. Formato de los paquetes DUNDi

Los paquetes enviados por el protocolo DUNDi presentan el siguiente formato:

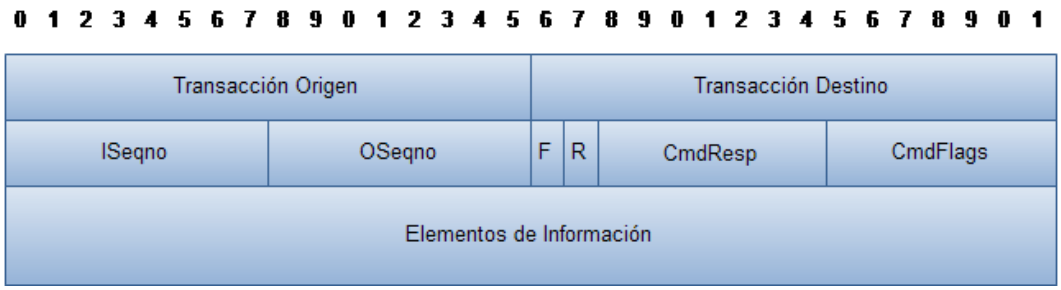


Figura 5.8.2: Formato del paquete del protocolo DUNDi.

- **Transacción origen:** Identificador único local de la transacción.
- **Transacción destino:** Identificador único remoto de la transacción.
- **ISeqno (Número de secuencia de entrada):** Representa el próximo valor esperado para el número de secuencia de entrada del próximo mensaje.
- **OSqno (Número de secuencia de salida):** Representa el número de secuencia de salida del mensaje en la transacción.
- **F:** Es el bit de finalización de transacciones, su valor es 1 cuando es el último mensaje de la transacción o es el ACK de la recepción de un mensaje de finalización.
- **R:** Este bit permite diferenciar si el mensaje se trata de un comando o una respuesta a un comando. El valor es 0 si se trata de un comando y 1 si es una respuesta. Solo los comandos pueden iniciar un dialogo o transacción, por ende los mensajes de apertura de dialogo deben enviar el mensaje con este valor en 0.
- **CmdResp:** Identifica de qué tipo de comando o respuesta se trata el paquete.
- **CmdFlags:** Bits de banderas específicas del comando enviado.
- **Elementos de información:** Este campo está compuesto por ninguno o varios elementos de información que son propios de los comandos o respuestas enviadas.

5.8.4. Comandos

Los comandos DUNDi están representados en los paquetes del protocolo por ocho bits, incluyendo los bits F y R. Cada comando posee un conjunto de elementos de información específicos del

comando, opcionales u obligatorios, necesarios para el procesamiento del comando y funcionamiento del protocolo DUNDi. Los elementos de información que se agreguen a un comando y que no pertenezcan a la especificación propuesta por la documentación de DUNDi deben ser ignorados durante el procesamiento del comando. Además, los comandos que no presenten los elementos de información obligatorios deben ser ignorados por los pares DUNDi.

El orden en que se presenten los elementos de información en el comando es irrelevante, excepto si está presente el elemento de información ENCDATA que siempre debe ser el último elemento.

Todos los comandos de respuesta deben establecer en 1 el valor del bit R.

Los comandos DUNDi son:

- **Respuesta ACK:** Este comando no requiere de elementos de información. Es utilizado para comunicar el arribo de un mensaje cuando no hay otros mensajes disponibles que enviar inmediatamente. El valor de número de secuencia de salida no debe incrementarse luego de enviar una respuesta ACK. La respuesta ACK no debe utilizarse como respuesta de otro ACK o de un mensaje INVALID.

- **Comando DPDISCOVER:** El propósito de este comando es iniciar el descubrimiento de un número dentro de un sistema DUNDi. Este comando generalmente se utiliza para iniciar una transacción y es respondido inicialmente con una respuesta ACK y luego con el comando de respuesta DPRESPONSE cuando el comando haya sido procesado por un nodo remoto. Los elementos de información de este comando son:
    - **VERSION (obligatorio):** Debe ser establecido con el valor 1 para la versión actual del protocolo.
    - **EID / EID_DIRECT (obligatorio):** Este elemento puede estar presente una o más veces en el comando. El primer elemento listado corresponde al identificador único del nodo que genera actualmente el requerimiento. El último elemento listado debe ser el identificador del nodo que generó originalmente el requerimiento.
    - **CALLED NUMBER (obligatorio):** El número solicitado que debe ser descubierto.

- **CALLED CONTEXT (obligatorio):** El contexto donde debe buscarse el número solicitado.
- **TTL (obligatorio):** El tiempo de vida del requerimiento.

- **Comando DPRESPONSE:** Este comando es de respuesta al comando DPDISCOVER y es utilizado para enviar la respuesta al nodo que origina la búsqueda de un número. Los elementos de información de este comando de respuesta son:
  - **CAUSE (opcional):** Puede indicar la causa de un fallo de una respuesta. Debe incluirse si se produce un error de autentificación.
  - **ANSWER (obligatorio):** Este elemento puede presentarse una o más veces en el comando, incluye todas las respuestas del sistema.
  - **HINT (obligatorio):** Indicaciones sobre las respuestas o la falta de los mismas.
  - **EXPIRATION (obligatorio):** Tiempo de expiración de la respuesta, la indicación o la falla.

- **Comando EIDQUERY:** El propósito de este comando es iniciar la búsqueda de información de un nodo DUNDi dentro del sistema a través de su identificador único (EID). Este comando generalmente inicia una transacción DUNDi, inicialmente es respondido con una respuesta ACK y luego con el comando de respuesta EIDRESPONSE luego de haber sido procesado el comando EIDQUERY por un nodo remoto. Los elementos de información de este comando son:
  - **VERSION (obligatorio):** Debe ser establecido con el valor 1 para la versión actual del protocolo.
  - **EID / EID_DIRECT (obligatorio):** Este elemento puede estar presente una o más veces en el comando. El primer elemento listado corresponde al identificador único del nodo que genera actualmente el requerimiento. El último elemento listado debe ser el identificador del nodo que generó originalmente el requerimiento.
  - **REQUESTED EID (obligatorio):** El identificador único solicitado que debe ser descubierto.

- **CALLED CONTEXT (obligatorio):** El contexto donde debe buscarse el identificador único solicitado.
- **TTL (obligatorio):** El tiempo de vida del requerimiento.

- **Comando EIDRESPONSE:** Este es el comando de respuesta del comando EIDQUERY que devuelve toda la información relacionada con el identificador único solicitado. Si el identificador fue encontrado, todos los valores de los elementos de información de este comando deben ser completados con los datos del nodo identificado excepto por el elemento de información IPADDR que es obtenido de contactarse directamente con el nodo identificado por el EID buscado. Los elementos de información de este comando son:
  - **DEPARTAMENT (opcional):** Departamento de contacto.
  - **ORGANIZATION (opcional):** Organización de contacto.
  - **LOCALITY (opcional):** Localidad o ciudad de contacto.
  - **STATEPROV (opcional):** Estado o provincia de contacto.
  - **COUNTRY (opcional):** País de contacto.
  - **EMAIL (opcional):** Email de contacto.
  - **PHONE (opcional):** Teléfono de contacto.
  - **IPADDR (opcional):** Dirección IP del nodo identificado por el EID solicitado, debe ser completado cuando se contacta al nodo buscado.
  - **HINT (obligatorio):** Indicaciones sobre las respuestas o sus fallos.

- **Comando INVALID:** Este comando de respuesta es enviado cuando un mensaje fue recibido fuera del alcance de una transacción DUNDi. Este comando puede ser enviado como respuesta de cualquier mensaje excepto un INVALID. Este mensaje no debe ser retransmitido a menos que se reciba otro en un estado inválido. Tampoco debe ser respondido por un mensaje ACK. Este comando no requiere de elementos de información.

- **Comando UNKNOWN:** El propósito de este comando de respuesta es indicar la recepción de comando desconocido o no implementado aún. El elemento de información de este comando es:

- **UNKNOWN (obligatorio):** El valor del comando que no fue reconocido.

- **Comando NULL:** Este comando de respuesta es utilizado para verificar la conectividad con un nodo o mantener una conexión abierta. No requiere de elementos de información adicionales.

- **Comando REGREQ:** El propósito de este comando es iniciar la registración de un nodo de un sistema DUNDi con otro nodo. Este comando generalmente inicia una transacción y es respondido inmediatamente con el comando de respuesta REGRESPONSE una vez el nodo remoto procesa el comando. Los elementos de información de este comando son:

  - **VERSION (obligatorio):** Debe ser establecido con el valor 1 para la versión actual del protocolo.
  - **EID (obligatorio):** Identificador único que será registrado en el sistema DUNDi. Este identificador generalmente es completado con la dirección MAC del nodo registrante.
  - **EXPIRATION (obligatorio):** Tiempo de vida máximo que posee la registración.

- **Comando REGRESPONSE:** El comando de respuesta al comando REGREQ para indicar si la registración se llevó a cabo satisfactoriamente o con error. Los elementos de información de este comando son:
  - **CAUSE (opcional):** Puede indicar la causa de un fallo de una respuesta. Debe incluirse si se produce un error de autentificación.
  - **EXPIRATION (obligatorio):** Tiempo máximo de validez de la registración antes de que sea renovada.

- **Comando CANCEL:** Este comando es utilizado para verificar la conectividad con un nodo o mantener una conexión abierta. No requiere de elementos de información adicionales. Para este comando el bit F debe ser establecido en 1.

- **Comando ENCRYPT:** El propósito de este comando es el de encriptar otro mensaje DUNDi incluyendo los elementos de información, para proveer autentificación y privacidad de la información de transacciones sensibles. El primer mensaje de encriptación debe contener el valor KEYCRC32 de referencia de la llave que será utilizada o enviar de forma implícita la llave con la que se encriptarán los mensajes. Los elementos de información de este comando son:
  - **EID (opcional):** Identificador único del nodo que envía el mensaje cifrado.
  - **KEYCRC32 (opcional):** CRC32 de las claves a ser utilizados para la encriptación.
  - **SHAREDKEY (opcional):** Llaves con las que se encriptarán los mensajes.
  - **SIGNATURE (opcional):** Firma RSA del contenido del elemento de información SHAREDKEY.
  - **ENCDATA (oblitagorio):** Información encriptada que es enviada a un nodo remoto. Siempre debe ser el último elemento de información del mensaje.

- **Comando ENCREJ:** Este comando de respuesta se utiliza para indicar que el mensaje cifrado (ENCRYPT) no fue recibido correctamente. El elemento de información de ese comando es:
  - **CAUSE (opcional):** Este elemento puede indicar la causa de la falla de la encriptación, la pérdida de la respuesta o error de autentificación.

5.8.5. Registración

Los comandos de registración son los que permiten a los nodos generar relaciones de confianza entre ellos, conocer su presencia en un sistema confiable y a partir de ello compartir contextos y directorios telefónicos. Estos comandos son REGREQ y REGRESPONSE, y son utilizados para conocer las direcciones dinámicas o traducidas de los nodos DUNDi. Un nodo envía el comando REGREQ a otro que responde con el comando REGRESPONSE independientemente de si acepta o no la registración. Si la registración es aceptada, esta debe ser renovada antes de que se cumpla el tiempo de expiración enviado en el campo EXPIRE en el comando REGRESPONSE si quiere mantener la registración.

5.8.6. Descubrimiento de directorios

El descubrimiento de directorios se realiza a través del comando DPDISCOVER y el comando de respuesta DPRESPONSE. El primero de los comandos generalmente inicia una transacción para realizar una consulta (el número a buscar y el contexto donde debería ser encontrado), si es un nodo autorizado para consultar recibe como respuesta primero un mensaje ACK y luego el comando DPRESPONSE con el bit F establecido en 1.

Un nodo ante la llegada de un comando DPDISCOVER debe responder con el comando DPRESPONSE dentro de T milisegundos, donde T es el resultado de multiplicar por 200 y sumarle 2000 al valor del elemento de información TTL del mensaje DPDISCOVER recibido. La entidad que envió el mensaje de descubrimiento debe terminar su requerimiento con el comando CANCEL si no recibe el comando DPRESPONSE luego de los T + 200 milisegundos de haber hecho el requerimiento. Los nodos que reciben el comando DPRESPONSE deben procesar y cachear la respuesta, para luego crear su propio comando DPRESPONSE a cualquier comando DPDISCOVERY que reciba con el mismo requerimiento realizado.

Si se envía un comando DPDISCOVER como resultado de haber recibido un DPDISCOVER de otra entidad, entonces:

1. El valor del elemento de información TTL de este mensaje debe ser el valor del elemento de información TTL del primer DPDISCOVER reducido en uno. Si el valor de TTL recibido del requerimiento original es cero, entonces no se debe enviar ningún requerimiento a otras entidades.

2. No se debe enviar el requerimiento a ninguna de las entidades que son listadas en el mensaje DPDISCOVER original.

3. El último elemento de información EID en el nuevo comando DPDISCOVER debe ser necesariamente el mismo que sea el último en el requerimiento original. El primer elemento de información EID debe ser el EID de la entidad que envía el nuevo comando DPDISCOVER. Por otro lado, todos los EID listados en el comando DPDISCOVER original

deben ser incluidos en cualquier nuevo comando DPDISCOVER que se realice.

Sin importar el origen del comando DPDISCOVER, para la respuesta DPRESPONSE siempre es válido que:

1. El comando DPRESPONSE puede incluir una o más respuestas, pero no debe incluir respuestas con la misma tecnología y destino. Si la respuesta es una combinación de requerimientos que comparten la misma tecnología y destino, sólo debe incluirse la respuesta que tenga el costo mínimo.

2. Si cualquiera de las entidades consultadas tenía el valor de TTL mayor a 1, entonces la respuesta debe establecer el valor de la bandera TTLEXPIRED.

5.9. Asterisk Realtime Architecture

Asterisk Realtime Architecture (voip-info.org, 2010) es una de las más importantes características de Asterisk a partir de la versión 1.2.

Por defecto, toda la configuración que hace parte de Asterisk, ya sea dialplans (planes de marcado), usuarios SIP, usuarios IAX, extensiones, casillas de correo de voz (voicemail), etc., se realizan sobre archivos de configuración de extensión .conf que se encuentran en el directorio /etc/asterisk de Linux. Cada vez que se modifica alguna configuración, los cambios deben aplicarse sobre Asterisk recargando los archivos de configuración o reiniciando el servicio. Con Asterisk Realtime todas estas configuraciones se mantienen en una base de datos (como MySQL o PostgreSQL).

Existen dos tipos de configuraciones Realtime. En una de ellas cada archivo de configuración es representado por una tabla, donde la mayoría de los parámetros específicos de configuración son representados por columnas en dicha tabla. De esta manera, puede lograrse que las configuraciones se puedan modificar durante la ejecución de los servicios y que los cambios sean reflejados al instante sin la necesidad de recargar las configuraciones o reiniciar los servicios. En la otra, los archivos de configuración son enteramente almacenados en una tabla donde cada registro de la tabla representa una línea de configuración del archivo convencional de

configuración. Estos registros son leídos de la base de datos únicamente durante el inicio o reinicio del servicio y por lo tanto las modificaciones son efectivas cuando esto sucede.

Con Asterisk Realtime Arquitecture se puede centralizar la configuración de varios servidores o, por ejemplo, de un clúster de servidores Asterisk, donde es necesario e importante mantener una configuración consistente y equivalente en todos los servidores. Por ejemplo, en el caso de tener que agregar un usuario o una nueva extensión con Realtime solamente es necesario crear una entrada en las tablas de usuarios o extensiones en la base de datos. De otra manera, hubiera sido necesario modificar cada uno de los archivos de configuración en todos los servidores teniendo la posibilidad de generar inconsistencias entre las configuraciones, y por lo tanto un mal funcionamiento del servicio. Además, esta característica brinda escalabilidad al compartir la misma configuración entre los servidores ya que no hace falta replicar las configuraciones al agregar un nuevo servidor al clúster.

5.9.1. Configuraciones de Asterisk Realtime

Si Asterisk requiere conectarse continuamente a la base de datos por cada paso que debe realizar en el dialplan, la base de datos será sobrecargada. Por esta razón, la configuración puede realizarse de dos maneras:

- **Realtime estático:** En este caso la información de la base de datos es cargada al iniciar el servicio o en cada reinicio. La ventaja que eso supone es que la carga sobre la base de datos es baja y si la base de datos falla luego de haber sido cargada no afecta a la ejecución del servicio. La desventaja que se presenta es que esta opción no es realmente en tiempo real, ya que si se realizan modificaciones en la base de datos se debe reiniciar Asterisk para que tome los cambios.

  En este tipo de configuración se almacenan los archivos regulares de configuración de Asterisk (.conf) en la base de datos.

  Todos los archivos de configuración del directorio /etc/asterisk que son almacenados en la base de datos deben ser removidos o renombrados para no generar alteraciones en la configuración de Realtime.

- **Realtime dinámico:** En este tipo de configuración lo que se almacena en la base de datos son los parámetros de configuración y los valores de los archivos de configuración de Asterisk. En esta opción, todos los cambios realizados en la base de datos se ven reflejados en tiempo real, pero si la base de datos falla también lo hará el servicio.

5.9.2. Archivos de configuración

Los archivos de configuración necesarios para poder llevar a cabo el uso de Realtime con Asterisk son los siguientes:

- **extconfig.conf:** Este archivo contiene toda la información necesaria para vincular la configuración de un servicio o característica de Asterisk con la base de datos, el tipo de conector de base de datos necesario y la tabla en donde se encuentra dicha configuración. Esta configuración es leída por Asterisk en el inicio y los reinicios, como así también cuando ciertos módulos son reiniciados (por ejemplo: sip reload).

  Para una configuración Realtime dinámico, la estructura de este archivo es la siguiente:

```
[settings]

<family name> =><driver>,<database name>,table_name
sippeers => mysql,asterisk,sip_peers
sipusers => mysql,asterisk,sip_users
queues => mysql,asterisk,queue_table
```

  En este caso los family name soportados son:

  - sippeers
  - sipusers
  - iaxpeers
  - iaxusers
  - voicemail
  - musiconhold
  - queues y queue_members (usados en conjunto para encolamiento de llamadas)
  - extensión

  El parámetro driver especifica el conector que debe utilizarse, ya sea MySQL u ODBC para

tomar la configuración desde los archivos res_mysql.conf o res_odbc.conf respectivamente para conectarse a la base de datos.

Si se omite el parámetro table_name por defecto se interpreta que el nombre de la tabla es idéntico al family name.

Para una configuración Realtime estático la estructura de este archivo es la siguiente:

```
[settings]

<conf filename> =><driver>,<databasename>,table_name
queues.conf => odbc,asterisk,ast_config
sip.conf => mysql,asterisk
```

A diferencia de la configuración anterior, en este caso se indica en que base de datos y tabla se encuentra la configuración correspondiente al archivo conf de Asterisk.

- **res_odbc.conf:** Archivo de configuración con los parámetros específicos para conectarse a la base de datos a través de un conector ODBC previamente configurado.

```
[asterisk]
enabled = yes
dsn = asterisk-cdr
username = asterisk
password = contraseña
loguniqueid = yes
pre-connect = yes
```

El parámetro dsn establece el nombre del conector ODBC previamente configurado para acceder a la base de datos. En el conector se establecen el usuario, password, nombre de base de datos, puerto y todo parámetro necesario para acceder a la base de datos donde se almacenará la configuración de Realtime.

El usuario y password que se especifiquen aquí deben coincidir con los establecidos en el conector.

La sección [asterisk] establece el nombre que se utilizará dentro del archivo extconfig.conf para hacer referencia al conector creado para acceder a la base de datos.

- **res_mysql.conf:** Archivo de configuración para conectarse a una base de datos MySQL.

```
[general]
dbhost = localhost
dbname = asterisk
dbuser = asterisk
dbpass = yourpassword
dbport = 3306
dbsock = /var/lib/mysql/mysql.sock
```

- **res_ldap.conf:** Archivo de configuración para utilizar Realtime LDAP. Los archivos de configuración son accedidos a través de directorios LDAP.

5.9.2.1. Ejemplo de configuración Realtime estático

Suponiendo que se requiere generar una configuración Realtime estático del archivo extensions.conf donde se mantienen todos los planes de marcado, el primer paso a realizar es establecer la tabla en base de datos donde se almacene dicha configuración. Esta tabla, de nombre ast_config por ejemplo, debe tener una estructura como la siguiente:

Columna	Descripción
Id	Identificador único para diferenciar los registros contenidos en la tabla de configuración.
Cat_metric	Establece el peso de la categoría especificada en la columna **category**. De esta manera se establece el orden de las categorías dentro del archivo de configuración.
Var_metric	Establece el peso de la opción o variable establecida en el campo **var_name**. De esta manera se establece el orden de los parámetros de configuración dentro de la categoría.
Filename	Nombre del archivo de configuración al que corresponde el registro, por ejemplo: iax.conf, sip.conf, etc.

category	Nombre de sección dentro del archivo de configuración, por ejemplo [general], pero sin la necesidad de almacenarlo con los corchetes.
Var_name	Nombre de la variable u opción que está siendo especificada en la línea de configuración del archivo. Por ejemplo, de la opción *disallow=all* en esta columna se almacena el valor *disallow*.
Var_val	El valor establecido para la variable u opción de configuración perteneciente a la variable **var_name**. Por ejemplo, de la opción *disallow=all* en esta columna se almacena el valor *all*.
Commented	Todo valor distinto de cero es evaluado como si fuera un comentario en el archivo de configuración

Tabla 5.9.1: Estructura de tabla para Realtime estático.

Suponiendo que el contenido del archivo extensions.conf es la que se presenta a continuación:

```
[general]
static=yes

[globals]
CONSOLE=Console/dsp ; Console interface for demo

[default]
exten=_.,1,SetVar(extension=${EXTEN})
exten=_.,2,Macro(dial_agi_ver.1.0.0,${EXTEN},${channel})
exten=h,1,Goto(done,${extension},1)
```

Los registros dentro de la tabla de la base de datos se deben ingresar respetando el orden de las categorías, esto se establece a través de la columna **cat_metric**. Además, debe respetarse el orden de las líneas de configuración de cada categoría. Para ello, todas estas líneas poseen el mismo valor para la columna **cat_metric** pero incrementando el valor de la columna **var_metric**. El resultado es el siguiente:

```
INSERT INTO `ast_config` (`cat_metric` , `var_metric` , `filename` ,
`category` , `var_name` , `var_val` ) VALUES ('0','0','extensions.conf',
'general', 'static', 'yes');

INSERT INTO `ast_config` (`cat_metric` , `var_metric` , `filename` ,
`category` , `var_name` , `var_val` ) VALUES ('1','0','extensions.conf',
'globals', 'Console/dsp', 'yes');

INSERT INTO `ast_config` (`cat_metric` , `var_metric` , `filename` ,
`category` , `var_name` , `var_val` ) VALUES ('2','0','extensions.conf',
'default', 'exten', '_.,1,SetVar(extension=${EXTEN})');

INSERT INTO `ast_config` (`cat_metric` , `var_metric` , `filename` ,
`category` , `var_name` , `var_val` ) VALUES ('2','1','extensions.conf',
'default', 'exten', '_.,2,Macro(dial_agi_ver.1.0.0,${EXTEN},${channel})');

INSERT INTO `ast_config` (`cat_metric` , `var_metric` , `filename` ,
`category` , `var_name` , `var_val` ) VALUES ('2','2','extensions.conf',
'default', 'exten', 'h,1,Goto(done,${extension},1)');
```

Luego, hay que indicarle al Asterisk como debe obtener los datos de esta tabla. Esto se logra modificando el archivo **extconfig.conf** con el siguiente contenido:

```
extensions.conf => odbc,asterisk,ast_config
```

5.9.2.2. Ejemplo de configuración Realtime dinámico

Para realizar una configuración Realtime dinámico, debe crearse una tabla en la base de datos por cada archivo de configuración donde las columnas corresponden a la mayoría de los parámetros u opciones de configuración posibles para dicho archivo.

Considerando el ejemplo anterior, la tabla correspondiente al archivo **extensions.conf** debe poseer las siguientes columnas:

Columna	Descripción
Id	Identificador único para diferenciar los registros contenidos en la tabla de configuración.
context	Establece a que contexto pertenece el parámetro u opción de configuración contenido en el registro.
exten	Establece el patrón o número de extensión correspondiente al plan de marcado.

priority	Establece el orden o prioridad en que deben ejecutarse las operaciones sobre la extensión establecida en la columna **exten**.
app	Acción, función de Asterisk o script externo que debe ejecutarse para la extensión en la prioridad establecida. Solo se especifica el nombre de la función a ejecutar.
appdata	En esta columna se establecen los parámetros que correspondan a la función que debe ejecutarse en esta prioridad.

Tabla 5.9.2: Estructura para extensiones en Realtime dinámico.

La configuración de las extensiones se logra ejecutando los siguientes inserts a la base de datos:

```
INSERT INTO `extensions` VALUES(1,'default','_.',1,'SetVar','
extension=${EXTEN}');
INSERT INTO `extensions` VALUES(2,'default','_.',2,'Macro','
dial_agi_ver.1.0.0,${EXTEN},${channel}');
INSERT INTO `extensions` VALUES(3,'default','h',1,'Goto','
done,${extension},1');
```

En el archivo **extconfig.conf** se establece la siguiente configuración para que Asterisk pueda resolver los planes de marcado obteniéndolos de la tabla:

```
extensions => odbc,asterisk,extensions
```

Y por último dentro del archivo **extensions.conf** colocamos lo siguiente:

```
[general]
static=yes

[globals]
CONSOLE=Console/dsp ; Console interface for demo

[default]
switch => Realtime/@extensions
```

La sentencia *switch* es la que permite que Asterisk conozca donde debe ir a buscar la información que se encuentra fuera del archivo de configuración convencional y resolver el plan de marcado de cada contexto, en este caso el contexto *[default]*. La sintaxis completa de esta sentencia permite entre otras cosas cambiar el contexto en donde debe buscar los datos en la base de datos,

establecer o cambiar el *family name* utilizado para la configuración de Realtime y establecer otras opciones adicionales. La sintaxis completa es:

```
switch => Realtime/[context]@[family][/options]
```

5.10. Resúmen

En este capítulo se presentaron varios conceptos y herramientas que resultan de importancia para la implementación de un sistema Asterisk distribuido y de alta disponibilidad.

Primero se mencionaron algunos conceptos básicos sobre sistemas distribuidos y se mostraron las principales características de dos de estos, como son los Clusters y Grids. Luego se abarcó también el concepto de alta disponibilidad y en forma resumida se mencionaron algunas de las principales configuraciones posibles.

Se abordó acerca de las formas de implementar alta disponibilidad en sistemas GNU/Linux, a través del uso de Heartbeat y Pacemaker. Se presentó en detalle la arquitectura de Heatbeat y su funcionamiento y luego se explicó el uso de Pacemaker ya que éste resulta ser una herramienta fundamental para la configuración de los servicios que se quieren prestar.

Por otro lado, también se habló de LVS, un proyecto que incluye en el Kernel Linux el módulo IPVS, el cual permite realizar balanceo de carga a nivel transporte. Se explicaron las tres formas de funcionamiento que éste permite, dónde se debe prestar especial atención a las ventajas que el Encapsulamiento IP presenta sobre las otras dos. Finalmente, se introdujeron dos herramientas que permiten facilitar a nivel usuario el uso y configuración de este módulo. Estas son Ldirectord e IPVSadm.

Por último, en este capítulo se hizo mención a varias infraestructuras posibles de implementación de Asterisk, donde se resaltó las ventajas que presenta un clúster de servidores híbridos y cómo esta puede implementarse a través del uso de DUNDi y Realtime. Estas dos fueron explicadas en detalle ya que resultan ser indispensables para la infraestructura que se desea implementar para este proyecto.

Capítulo 6: Casos Prácticos de Calidad de Servicio

6.1. Introducción

En base a la investigación y la teoría expuesta se realizaron distintos casos prácticos para poner a prueba los conocimientos obtenidos. La práctica de Calidad de Servicio fue desarrollada en Linux bajo la arquitectura USER MODE LINUX (UML).

Al utilizar máquinas virtuales en lugar de equipos reales, existen determinados detalles a tener en cuenta para que los resultados de los experimentos realizados puedan considerarse válidos y ayuden a arribar a conclusiones útiles. Uno de estos detalles es la velocidad máxima de transferencia de los enlaces virtuales. Al tratarse de procesos de una computadora, la velocidad mencionada se encuentra altamente determinada por la carga actual del sistema, por lo tanto, para poder garantizar que estas velocidades se cumplan y no varíen significativamente se utilizan valores relativamente pequeños en relación a las velocidades máximas que pueden ser emuladas. Luego de distintas pruebas y mediciones, las velocidades máximas alcanzadas en el entorno virtual para esta tesis oscilaron alrededor de los 9 Mb/s. Por esta razón, durante las pruebas se limitó el ancho de banda entre los 32 y 256 kbit/s para garantizar que los objetivos de mismas se cumplan sin que la carga del procesador afecte los resultados. Por otro lado, se intentó mantener la carga al mínimo posible para que las condiciones en qué se ejecutó cada una de las pruebas sean las mismas.

6.2. Servicios diferenciados

Se diseñó un escenario, visualizado por la Figura 6.2.1, compuesto de tres dominios, donde se albergaron distintos Host. Éstos fueron los encargados de enviar flujos de distinto tipo de un dominio a otro, entre ellos un flujo de VoIP el cual será el más importante de todos.

El escenario fue puesto en marcha en un principio sin ninguna configuración de QoS. Únicamente se configuró una limitación de ancho de banda en los enlaces para poder experimentar cuellos de botella y congestión entre los dominios. A está corrida del escenario se la referenciará

posteriormente como ***Caso Base***. Luego se realizaron las configuraciones correspondientes para priorizar, en base al campo DSCP, ciertos flujos sobre otros y beneficiar a aquellos que se consideren más importantes según sus requerimientos (en tiempo real, bajo retardo, eficiencia, etc.). Esta ejecución del escenario será referenciada como ***Caso QoS***. Por último, se protegió todo el tráfico entre los dominios A y B, para garantizar seguridad al pasar por un dominio inseguro (dominio C). Se referenciará a esta prueba como ***Caso QoS/Seguro***.

El análisis de las pruebas consiste de los siguientes conceptos:

Parámetros de evaluación	
Duración de la transmisión	Tiempo transcurrido desde el inicio de la transmisión hasta el final de la conexión.
Ancho de banda	Asignación de ancho de banda del total disponible al flujo transmitido.
Paquetes transmitidos	Cantidad total de paquetes analizados en la captura.
Priorización de flujos	Nivel de importancia del flujo ante situaciones de congestión.

Tabla 6.2.1: Parámetros de evaluación para QoS.

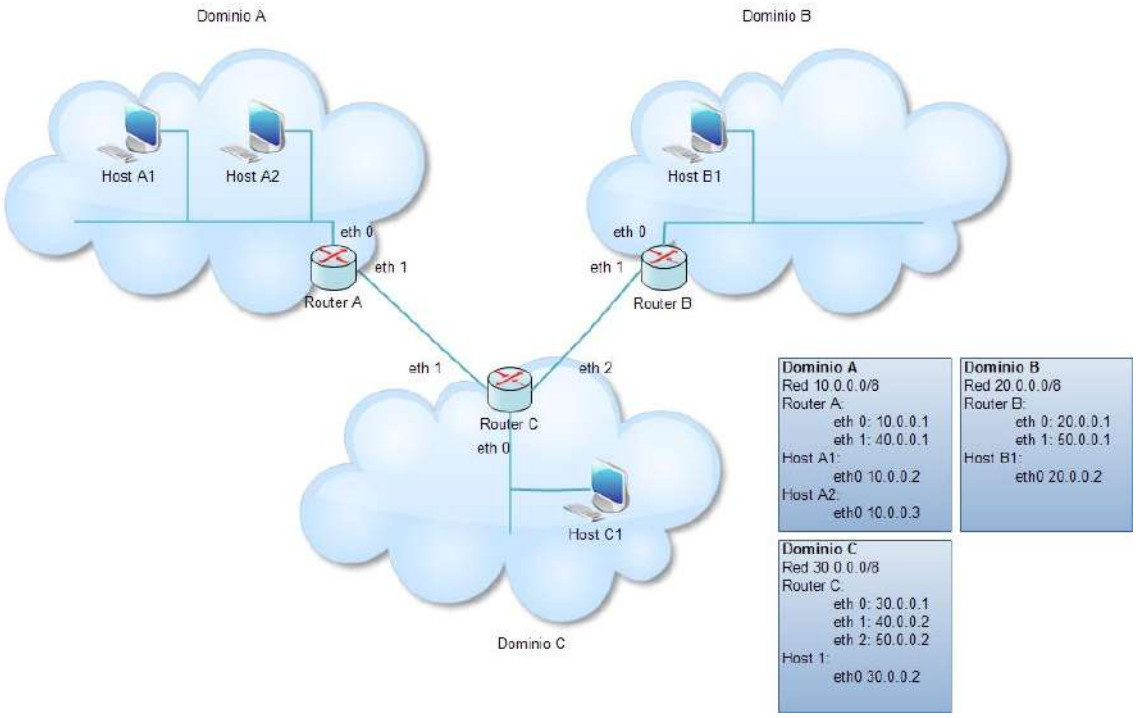


Figura 6.2.1: Escenario DSCP.

6.2.1. Herramientas

- **Tc:** Conjunto de mecanismos y operaciones por los cuales los paquetes son encolados en una interface de red para su trasmisión o recepción.
- **Tcpdump:** Herramienta que permite capturar y mostrar en tiempo real los paquetes que son transmitidos por la red.
- **IPsec-Tools:** Conjunto de herramientas para la implementación de IPsec.
    - **Setkey:** Herramienta que permite manipular y limpiar las base de datos SPD y SAD.
    - **Racoon:** Demonio que permite realizar el intercambio automático de claves de seguridad.
- **JTG:** Generador de tráfico de red que permite enviar paquetes de características específicas que permiten emular flujos reales de red.

6.2.2. Configuraciones

Para que los routers apliquen QoS sobre los flujos que viajen a través de ellos es necesario utilizar la herramienta TC mencionada en el Capítulo 3:. La configuración de los routers se encuentra en el Apéndice A. Dicha configuración, Figura 6.2.2, crea una Qdisc raíz de disciplina DSMARK, la cual copiará el valor del campo DSCP de los paquetes a una estructura de datos interna del kernel de Linux. Por debajo de esta disciplina se encuentra una Qdisc intermedia del tipo HTB que se encargará de repartir el total del ancho de banda del enlace. Finalmente, se encuentran las Qdisc hojas que se encargarán de planificar la salida de los paquetes pertenecientes a los grupos EF, AF1, AF2, AF3 y AF4. El flujo de paquetes marcados con EF será administrado por una disciplina *pfifo* con un ancho de banda garantizado de 128 Kbits y un máximo de 256 Kbits. Los flujos asignados a las clases AF1, AF2, AF3, y AF4 poseerán un ancho de banda de 64 Kbits, 32 Kbits, 19.2 Kbits y 12.8 Kbits respectivamente. Cada una de las clases será administrada por una disciplina GRED, que poseerá tres bandas para manejar los tres niveles de caída de precedencia (*Drop Precedence*). Los filtros establecidos en la configuración se encargan de realizar las operaciones necesarias para reconocer el campo de DSCP en los paquetes y enviar cada clase a la disciplina de cola correspondiente.

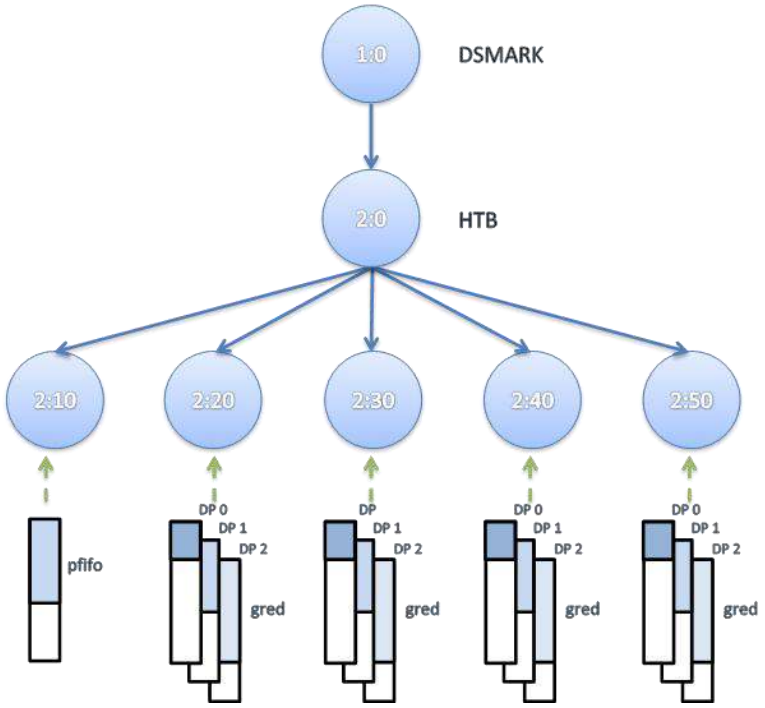


Figura 6.2.2: Disciplinas de cola para las pruebas.

6.2.3. Configuración de IPsec

Para la implementación del escenario lo primero que se realizó fue preparar el kernel para que brinde soporte de IPsec. Las opciones a activar son las siguientes:

```
Networking support (NET) [Y/n/?] y
  *
  * Networking options
  *
  PF_KEY sockets (NET_KEY) [Y/n/m/?] y
IP: AH transformation (INET_AH) [Y/n/m/?] y
  IP: ESP transformation (INET_ESP) [Y/n/m/?] y
  IP: IPsec user configuration interface (XFRM_USER) [Y/n/m/?] y

Cryptographic API (CRYPTO) [Y/n/?] y
  HMAC support (CRYPTO_HMAC) [Y/n/?] y
  Null algorithms (CRYPTO_NULL) [Y/n/m/?] y
  MD5 digest algorithm (CRYPTO_MD5) [Y/n/m/?] y
  SHA1 digest algorithm (CRYPTO_SHA1) [Y/n/m/?] y
  DES and Triple DES EDE cipher algorithms (CRYPTO_DES) [Y/n/m/?] y
  AES cipher algorithms (CRYPTO_AES) [Y/n/m/?] y
```

Si se trabaja con una versión del kernel inferior a la 2.6.28, como fue necesario para este trabajo, se debe parchear y volver a compilar para solucionar un bug en la generación de claves secretas. El código necesario para esto es el siguiente:

```
--- a/net/key/af_key.c
+++ b/net/key/af_key.c
@@ -2075,7 +2075,6 @@  static int pfkey_xfrm_policy2msg(struct sk_buff *skb,
struct xfrm_policy *xp, in
                          req_size += socklen * 2;
                 } else {
                          size -= 2*socklen;
-                         socklen = 0;
                 }
                 rq = (void*)skb_put(skb, req_size);
                 pol->sadb_x_policy_len += req_size/8;
```

La configuración de IPsec se realizó sobre el router A y el router B. Al ser completamente asimétricas, solo se incluirán las configuraciones del router A. Lo primero a realizar es agregar las entradas a la SPD para que el tráfico sea encriptado con el protocolo ESP en modo tunel. Esto se hace de la siguiente manera:

```
spdadd 10.0.0.0/8 20.0.0.0/8 any -P out ipsec
esp/tunnel/10.0.0.1-20.0.0.1/require;

spdadd 20.0.0.0/8 10.0.0.0/8 any -P in ipsec
esp/tunnel/20.0.0.1-10.0.0.1/require;
```

Posteriormente, debe configurarse a la herramienta racoon para que se encargue de la distribución de las claves.

Además, se necesita una clave secreta para autentificar al par racoon con el cual se realiza el intercambio de claves. Ésta debe ser la misma en cada extremo, en este caso almacenada en un archivo llamado sharedkey.txt. Es sumamente importante que este archivo sólo posea permisos de lectura y escritura únicamente para el usuario propietario, que debe ser únicamente el usuario root, y ningún otro permiso debe ser concedido.

```
path pre_shared_key "sharedkey.txt";

remote 20.0.0.1
{
exchange_mode main;
proposal
  {
    encryption_algorithm 3des;
    hash_algorithm md5;
    authentication_method pre_shared_key;
    dh_group modp1024;
  }
}

sainfo anonymous
{
  pfs_group modp768;
  encryption_algorithm 3des;
  authentication_algorithm hmac_md5;
compression_algorithm deflate;
}
```

6.2.4. Puesta en marcha de Diffserv e IPsec

Para obtener un mayor entendimiento de los casos de prueba realizados se pasará a explicar cómo es que funcionan Diffserv e IPsec utilizando las configuraciones antes mencionadas.

Una de las primeras cuestiones a resolver es la clasificación de los flujos en clases y determinar el nivel de importancia de los mismos en cada una de ellas. En el RFC 4594 “Configuration Guideline for Diffserv Service Classes” podemos encontrar un conjunto de codepoints sugeridos para los distintos tipos de tráfico de acuerdo a los requerimientos de la red (baja latencia, ancho de banda, etc.) y fue útil como referencia para llevar a cabo este propósito.

Cada router DS Ingress/Egress de los dominios del escenario fue configurado utilizando *iptables* para marcar los paquetes. El reconocimiento del flujo al que pertenece cada paquete se realizó de acuerdo al puerto destino o puerto fuente del header correspondiente a la capa de transporte (UDP/TCP). Esta configuración fue necesaria sobre la interface eth0, por ser la interface de salida de cada dominio. De esta manera los paquetes son enviados y recibidos por los dominios con las marcas correspondientes.

Por ejemplo, para marcar los paquetes que pertenecen al flujo VoIP que utilizan IAX2 como protocolo de señalización y transporte de datos debemos verificar que el puerto destino del

header del protocolo de transporte UDP contenga el valor 4569:

```
iptables -t mangle -A FORWARD -i eth0 -s $red_eth0 -p UDP --dport 4569 -j DSCP
--set-dscp 46
```

Cuando el paquete ingresa por la interface eth0 del router DS Egress podemos observar, en la Figura 6.2.3, que no posee ninguna marca y es despachado a la siguiente interfaz de salida.

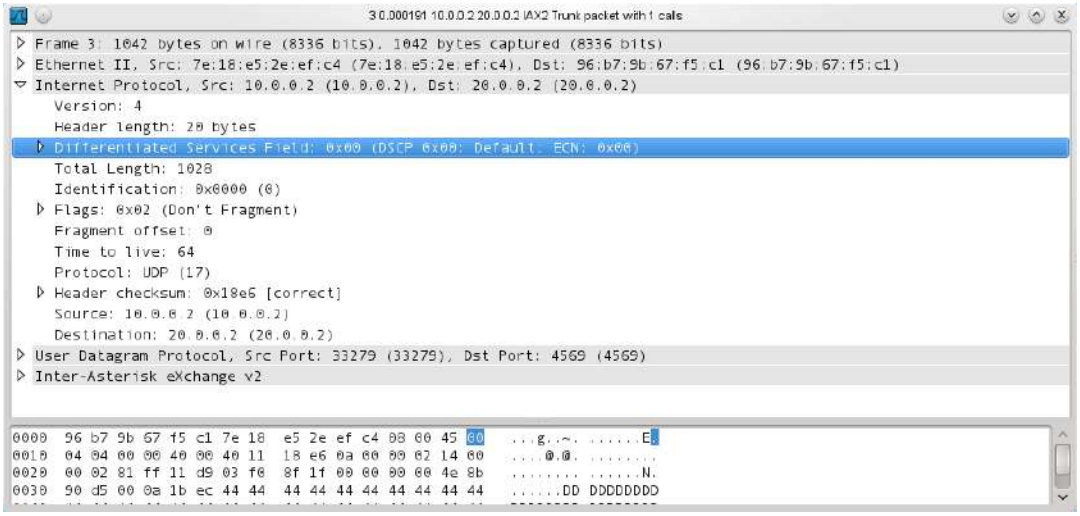


Figura 6.2.3: Paquete IAX2 sin marcar.

El paquete es encolado y despachado por la interface de salida correspondiente a la red destino, de acuerdo a la marca obtenida (ver Figura 6.2.4) al traspasar la interface eth0.

Una vez que los paquetes reciben el valor DSCP deben ser despachados a través de la cola creada especialmente para administrar un grupo AF o EF y darle el tratamiento correspondiente a dicha clase. Es aquí donde entran en funcionamiento los filtros aplicados mediante la herramienta *tc*. Su funcionamiento se explicará con el siguiente ejemplo, siguiendo la Figura 6.2.5.

Suponiendo que arriba un paquete perteneciente al grupo AF3 con precedencia de descarte 2, es decir AF32 (0x70 ó 01110000). El valor correspondiente a AF32 se encuentra en primer lugar dentro de la estructura de datos *skb->iph->tos*.

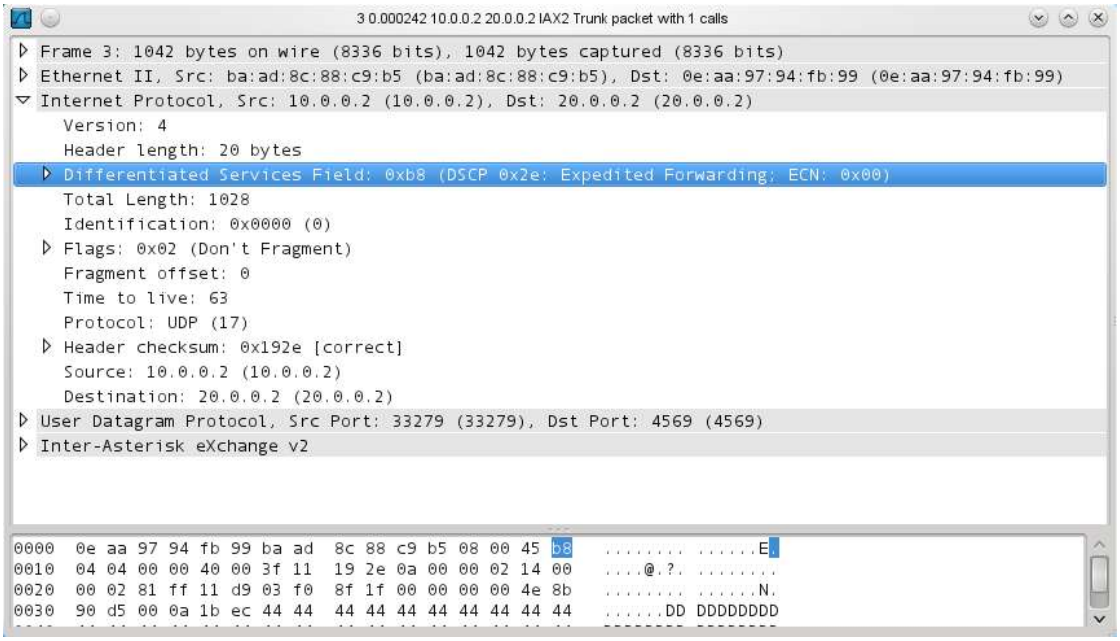


Figura 6.2.4: El mismo paquete IAX2 marcado.

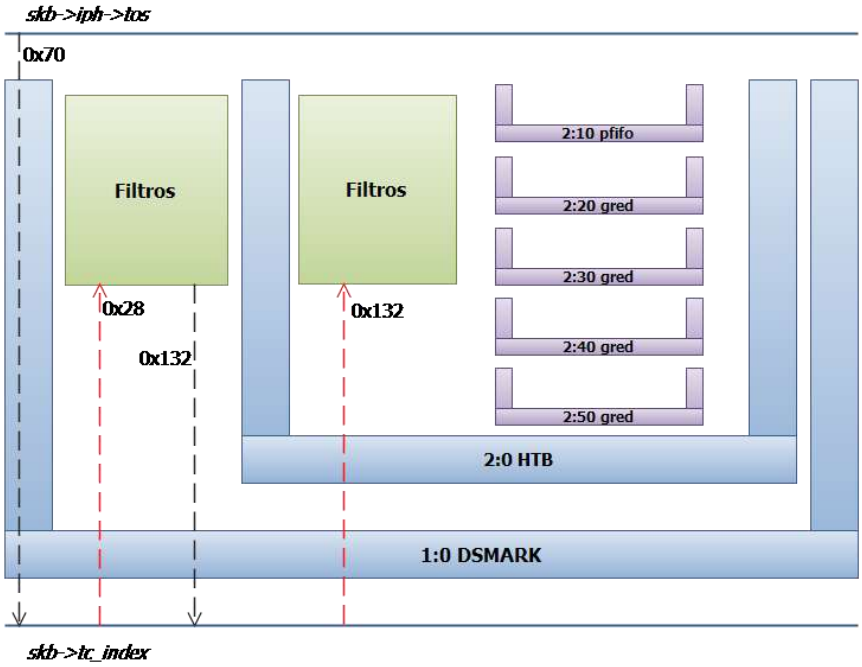


Figura 6.2.5: Seguimiento del paquete dentro de las disciplinas.

El primer conjunto de filtros se aplican sobre la disciplina de cola DSMARK, que como se mencionó se encarga **únicamente** de copiar el valor del campo DSCP a la estructura interna de Linux *sbk->tc_index.*

El primer filtro de este conjunto es el encargado de obtener el valor DSCP del campo TOS (*skb->iph->tos*) a través de la máscara 0xFC (11111100) y corriendo el resultado dos posiciones a la derecha. El valor de los dos últimos bits del campo DS corresponden a ECN y el contenido de éstos no interfieren con la asignación del paquete a una clase, es por esto que no son tenidos en cuenta durante el proceso y tampoco son copiados a la estructura *skb->tc_index*. El valor resultante de la operación se muestra a continuación:

```
  01110000
& 11111100
----------
  01110000  2 >>  00011100
```

Inmediatamente se aplica el siguiente filtro del conjunto que coincida con el valor obtenido del proceso anterior, en este caso el filtro que maneje el valor 28 (00011100) en decimal.

```
tc filter add dev eth0 parent 1:0 prio 1 handle 28 tcindex classid 1:131
```

Como se ve, el filtro asigna el paquete a la clase 1:131 que es inexistente dentro de la configuración. El valor *minor* de la clase es copiado a la estructura *skb->tc_index* en hexadecimal (0x131) y es valor a evaluar en los siguientes filtros. Este es un pequeño "truco" para poder determinar a través del siguiente grupo de filtros a que clase pertenece el paquete y la disciplina de cola final pueda determinar el valor de precedencia de descarte para dicho paquete.

El segundo conjunto de filtros se aplica sobre la disciplina de cola HTB y el resultado de los filtros para este conjunto no afecta y **no es copiado** dentro de la estructura *skb->tc_index*. Es muy común confundirse este último hecho cuando se utiliza en un nivel superior la disciplina DSMARK como también es fácil confundirse al pensar que lo que se modifica es el encabezado IP del paquete cuando se aplica una de estas máscaras.

El objetivo de este conjunto de filtros es determinar el grupo al que pertenece el paquete para que éste reciba el tratamiento correspondiente, que abarca el ancho de banda asignado a la clase y la disciplina de cola final que debe encargarse de despachar dicho paquete.

En este caso, el primer filtro del conjunto aplica la máscara 0xF0 y luego desplaza cuatro lugares a la derecha los bits del resultado. La operación es la siguiente:

```
  100110001
& 011110000
  ---------
  000110000   4 >>  000000011
```

Como resultado de la máscara se obtiene el valor 3 (00000011) en decimal. Este valor determina la clase del paquete y siguiendo la Tabla 6.2.2 podemos observar que el paquete pertenece a la clase AF3.

Valor	Clase
1	AF1
2	AF2
3	AF3
4	AF4
5	EF

Tabla 6.2.2: Correspondencia entre filtros y clases.

La clase de cola final utilizada para las pruebas es la disciplina GRED. Esta disciplina permite generar una colección de colas virtuales donde los paquetes son asignados a cada una de ellas de acuerdo al valor de precedencia de descarte que posea dicho paquete. Este valor se obtiene de los cuatro bits menos significativos (los últimos cuatro bits) de la estructura de datos *skb->tc_index*.

```
tc filter add dev eth0 parent 2:0 prio 1 handle 3 tcindex classid 2:40
```

Para el ejemplo, el último valor que toma la estructura es 0x132 que en binario corresponde al valor 100110001. Los cuatro bits menos significativos corresponden al valor 1 (0001) en decimal para la precedencia de descarte. Finalmente, el paquete será despachado por la disciplina GRED del grupo AF3 utilizando la cola virtual correspondiente al DP 1.

```
tc qdisc add dev eth0 parent 2:40 handle 40: gred setup DPs 3 default 2 grio

tc qdisc change dev eth0 parent 2:40 gred DP 0 limit 150000 min 20000 max
80000 avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.01 prio 1 # AF31

tc qdisc change dev eth0 parent 2:40 gred DP 1 limit 150000 min 20000 max
80000 avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.03 prio 2 # AF32

tc qdisc change dev eth0 parent 2:40 gred DP 2 limit 150000 min 20000 max
80000 avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.06 prio 3 # AF33
```

Vale aclarar que si bien durante la teoría de Servicios Diferenciados se explicó que los niveles de precedencia de descarte son del 1 al 3 por cuestiones de configuración estos valores fueron correlacionados a los valores del 0 al 2.

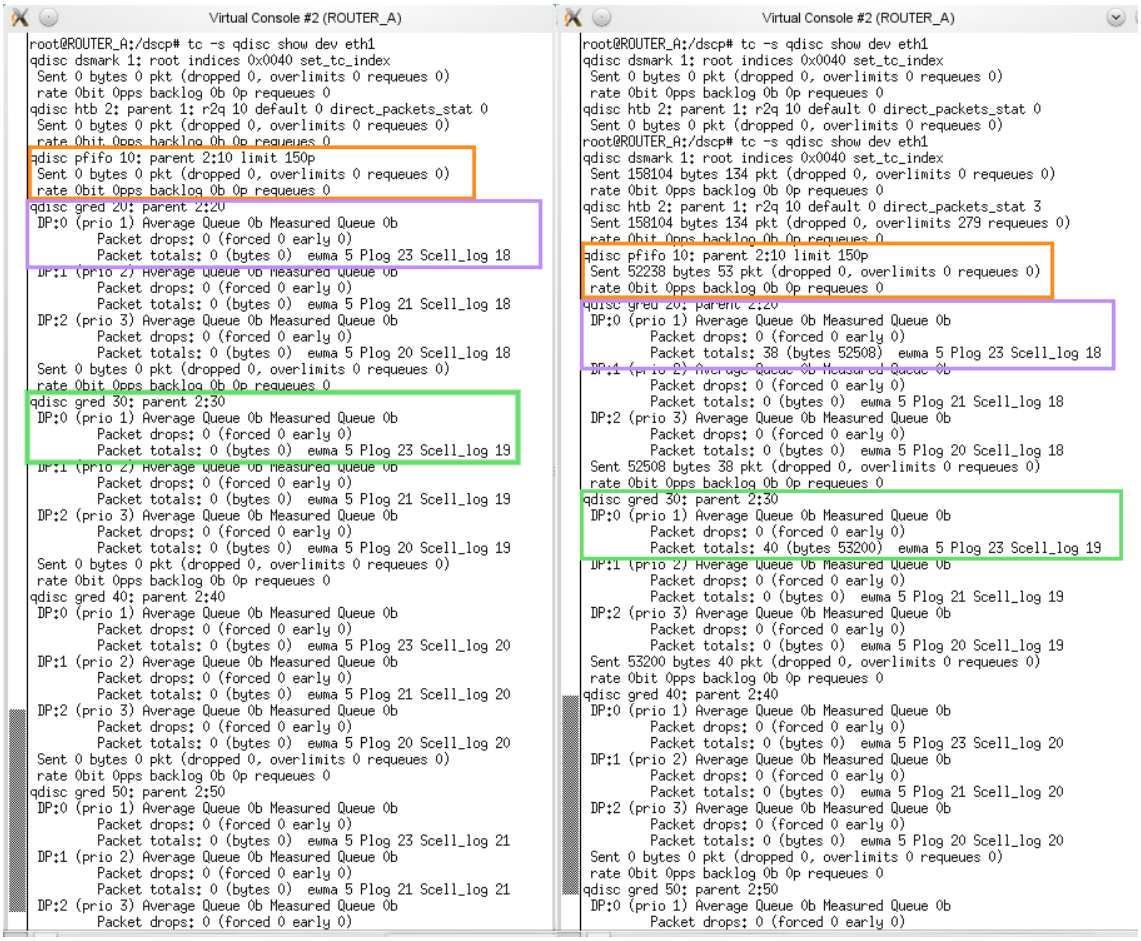


Figura 6.2.6: Estadísticas de paquetes encolados.

En la Figura 6.2.6 se muestran las estadísticas de *tc* antes (lado izquierdo) y después (lado derecho)

de enviar tráfico entre dominios. El recuadro color naranja corresponde al tráfico del grupo EF, el recuadro violeta muestra información de los paquetes encolados para la clase AF11, y finalmente el recuadro verde corresponde a la información relacionada a la clase AF21.

6.2.5. Pruebas y análisis de resultados

El objetivo de estas pruebas es mostrar el mejoramiento significativo que se obtiene en las transmisiones de información cuando los flujos son clasificados por alguna característica en especial, como puede ser flujos de streaming o VoIP, y tratados de forma preferencial según un nivel de importancia que se adecúe a las necesidades del dominio.

Cada caso de prueba fue ejecutado veinte veces, y los datos obtenidos se encuentran en los Apéndice B. Los resultados del ***Caso Base***, en la Figura 6.2.7, nos muestran que al no haber preferencias ni prioridades entre los distintos flujos, el ancho de banda fue repartido equitativamente entre ellos. De esta manera, la cantidad de paquetes recibidos/perdidos y la duración de la transmisión de los flujos fueron muy similares. Esto se debió a que los routers utilizan por defecto un PHB Best Effort.

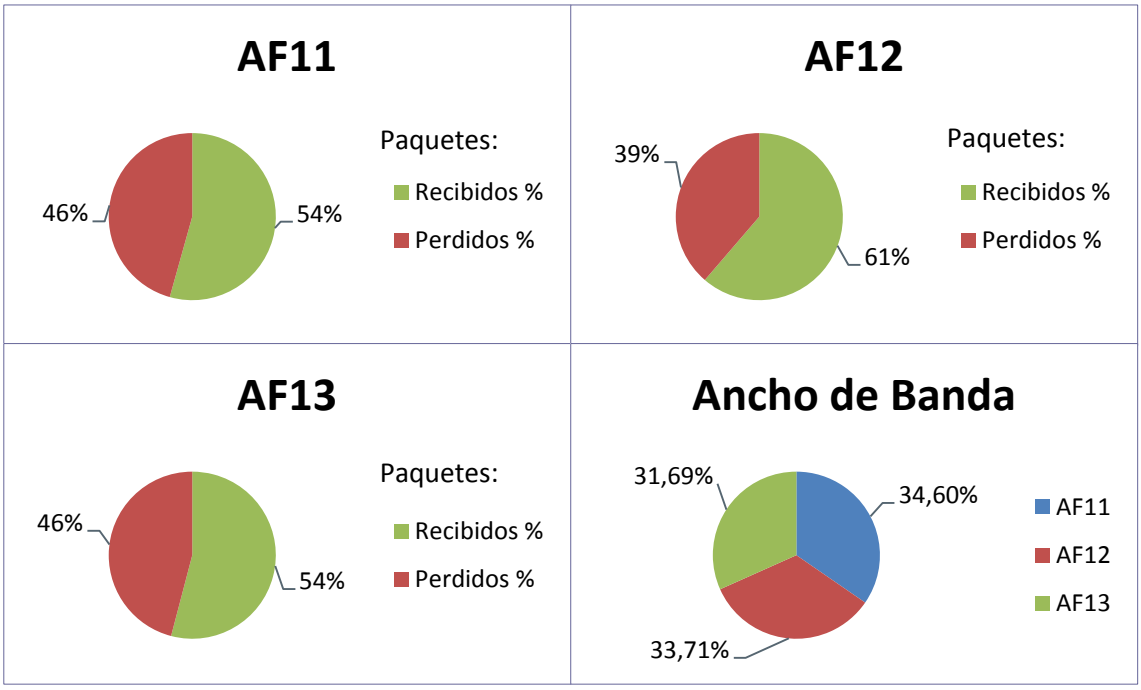


Figura 6.2.7: Resultados del caso Base.

El ***Caso QoS*** fue dividido en dos partes, y las denominaremos ***Caso QoS1*** y ***Caso QoS2.*** El primero se configuró para que los flujos sean clasificados como AF11, AF12, AF13, para observar el comportamiento de priorización según la caída de precedencia (*Drop Precedence*). En el segundo, se configuró para que los flujos sean clasificados como EF, AF11 y AF21, para analizar el tratamiento que recibirán los flujos cuando son asignados a distintas clases. Los primeros resultados se muestran en la Figura 6.2.8, mientras que los resultados de la segunda configuración se muestran en la Figura 6.2.9.

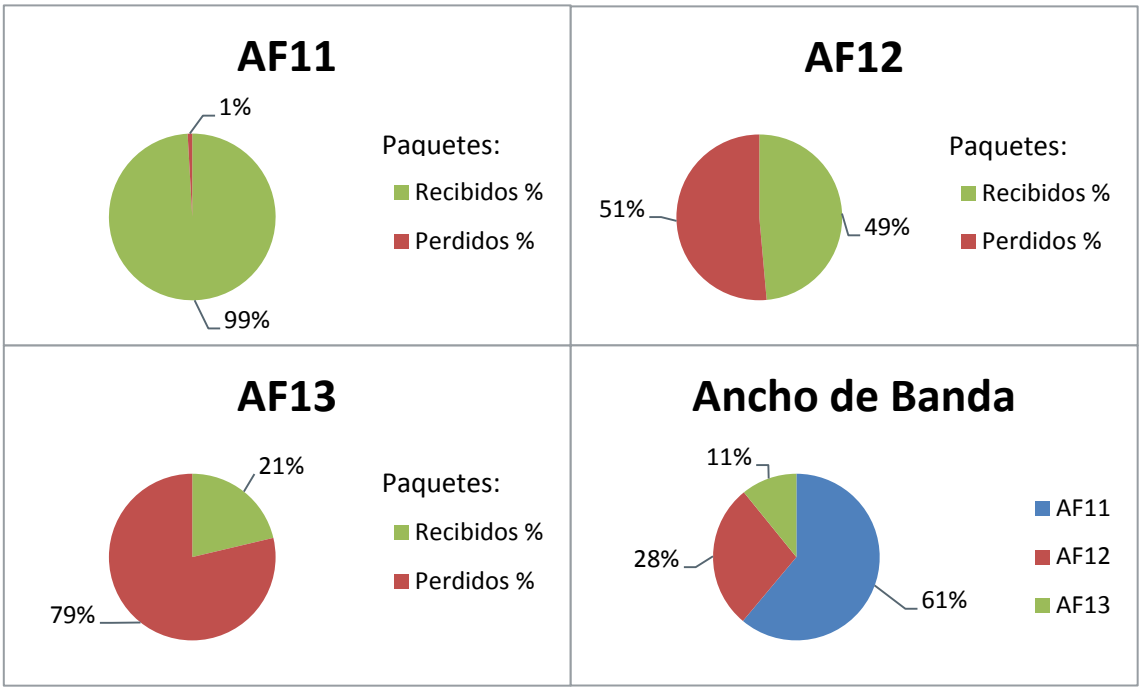


Figura 6.2.8: Resultados del caso QoS1.

Como se puede observar, se han obtenido resultados muy similares entre el ***Caso QoS1*** y el ***Caso QoS2***, pero muy distintos con respecto al ***Caso Base***. Éste último, repartía equitativamente el ancho de banda al no haber priorización, en cambio, en el ***Caso QoS*** el ancho de banda fue priorizado según la clase a la que pertenecía el flujo (***QoS2***) y según su caída de precedencia (***QoS1***). De esta manera, podemos tratar a los flujos de forma prioritaria asignándolos a una clase de servicio, y a su vez, dentro de la misma clase, podemos priorizar aún más haciendo uso de los bits de caída de precedencia.

Comparando el ***Caso Base*** con el ***Caso Qos1***, puede verse que la clase con mayor priorización (el flujo AF11) obtuvo una mayor asignación del ancho de banda y por ende aumentó el número de paquetes entregados. El flujo AF12, en cambio, no se diferenció tanto de la prueba anterior. La cantidad de paquetes perdidos se redujo solamente del 50% al 40% aproximadamente y tardó 3 segundos más en promedio en la entrega de los paquetes. El flujo de menor prioridad fue, lógicamente, el más perjudicado obteniendo resultados de calidad visiblemente inferiores producto del beneficio obtenido por el flujo AF11.

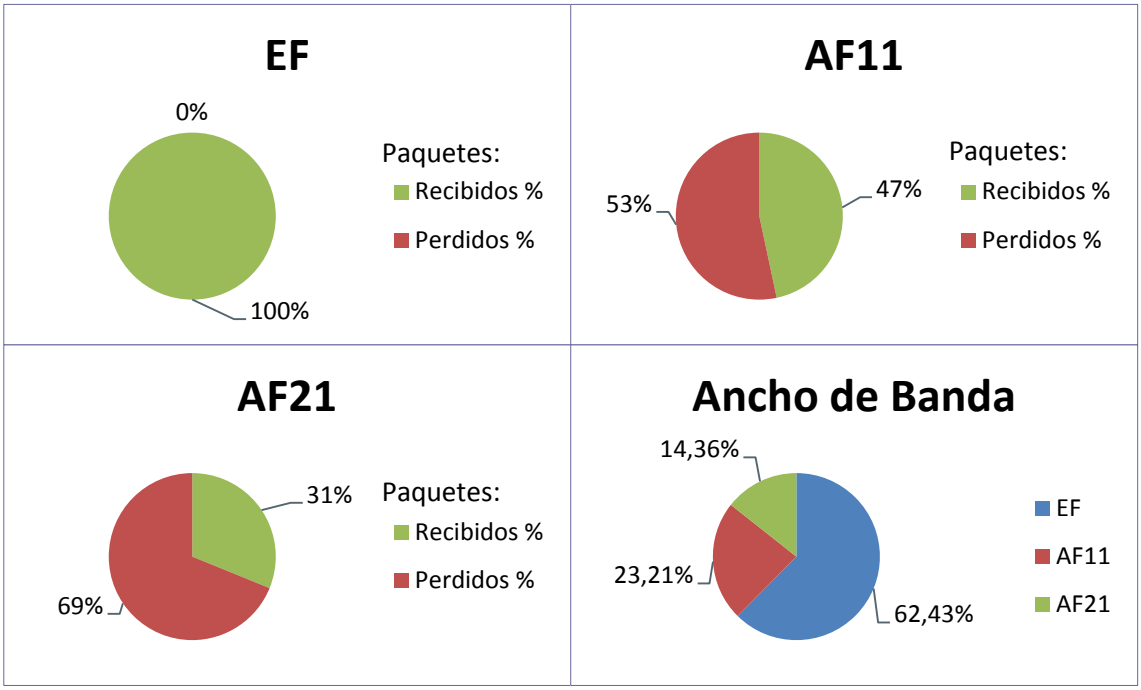


Figura 6.2.9: Resultados del caso QoS2.

Finalmente, resta analizar el efecto de añadir IPsec al escenario para proteger los flujos transmitidos entre los dominios de confianza A y B. En la Figura 6.2.10 se muestran los resultados del ***Caso QoS/Seguro***.

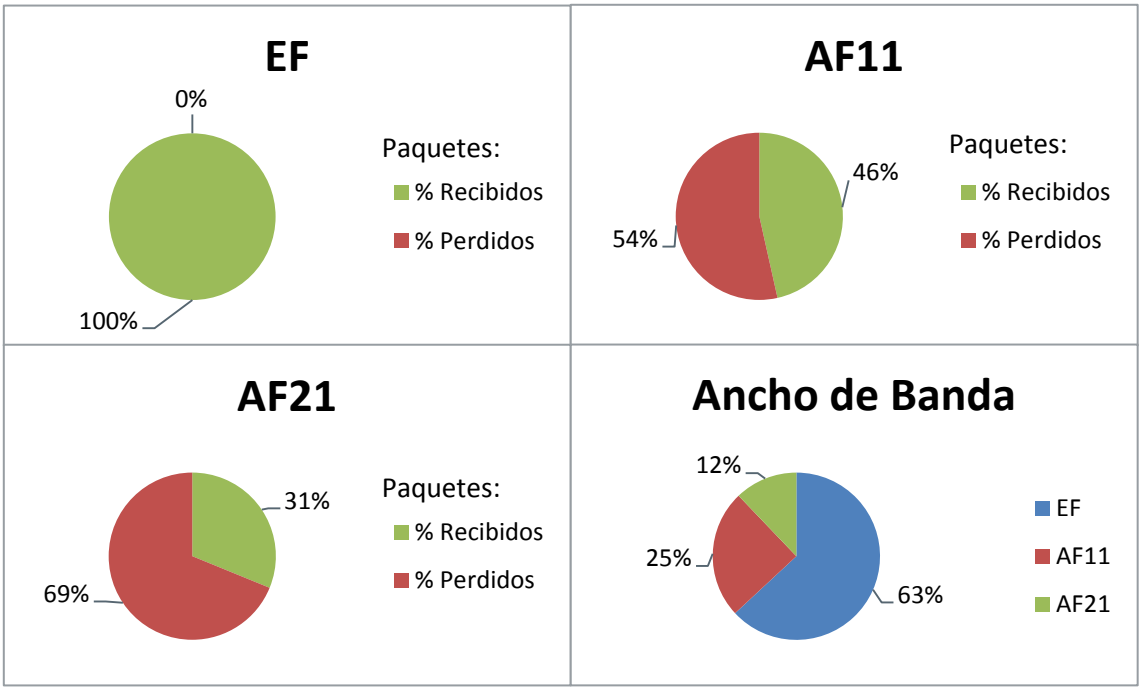


Figura 6.2.10: Resultados del caso Qos/Seguro.

Comparando con los resultados obtenidos y el ***Caso QoS2***, la inclusión de IPsec no degrado la performance del escenario. Esto es totalmente beneficioso, ya que además de darle mayor prioridad a los flujos deseados, podemos brindarles seguridad evitando así que terceros puedan inyectar tráfico malicioso, o simplemente capturar información sensible que estemos transmitiendo.

En cuanto a la complejidad de las configuraciones, es muy fácil cometer un error si es que no se ha planificado anteriormente un esquema de priorización donde ningún flujo de propósito para el proveedor quede excluido. La decisión de qué disciplina de cola utilizar está fuertemente ligada al esquema. Por ejemplo, de querer utilizar ECN sobre las configuraciones anteriores haría que la disciplina de cola final a utilizar sea RED en lugar de GRED, siendo esto un problema ya que RED no tiene en cuenta los valores de Drop Precedence al momento de descartar un paquete en situaciones de congestión y por lo tanto la utilización de clases y niveles no tendría razón de ser.

6.3. Notificación de Congestión Explícita (ECN)

Como se explicó en la sección 2.5, ECN es un mecanismo destinado a disminuir los periodos de congestión que pueden producirse en los routers de un dominio. Estas congestiones ocurren cuando el volumen de información entrante en los routers es mayor al volumen que éstos son capaces de procesar sin perjudicar su rendimiento.

Con el propósito de comprender mejor el funcionamiento de ECN y sus ventajas, se plantea un escenario compuesto de dos dominios separados entre sí por dos routers. Los enlaces dentro del domino poseen una velocidad de transmisión mayor al enlace que conecta a los routers extremos de cada dominio, de esta forma se asegurará que la red experimente congestión sobre esos routers cuando los dominios comiencen a intercambiar información.

El escenario, representado por la Figura 6.3.1, será configurado para utilizar políticas denominadas droptail y políticas que solucionan las desventajas de las anteriores. Para este trabajo fue elegida la disciplina Token Backet Filter (droptail) y la disciplina RED que como se explicó en la sección 3.4.2 surge como solución a las políticas droptail, siendo además de las pocas políticas que pueden ser utilizadas implementando ECN. Además, el escenario será evaluado utilizando dos implementaciones diferentes de algoritmos de control de congestión del protocolo TCP para ver como ECN influye en el control de congestión independientemente del algoritmo utilizado.

La mayor complejidad de la experiencia radica en obtener los valores de la ventana de congestión. Esto se debe a que son valores que se mantienen en la memoria RAM del emisor y no son transmitidas al receptor. Las herramientas gratuitas que pueden encontrarse en Internet por el momento no ofrecen la entrega de este dato, lo que suele realizarse es aproximar dicho valor mediante un cálculo utilizando el tiempo transcurrido entre la emisión de un paquete y la recepción de su ACK correspondiente.

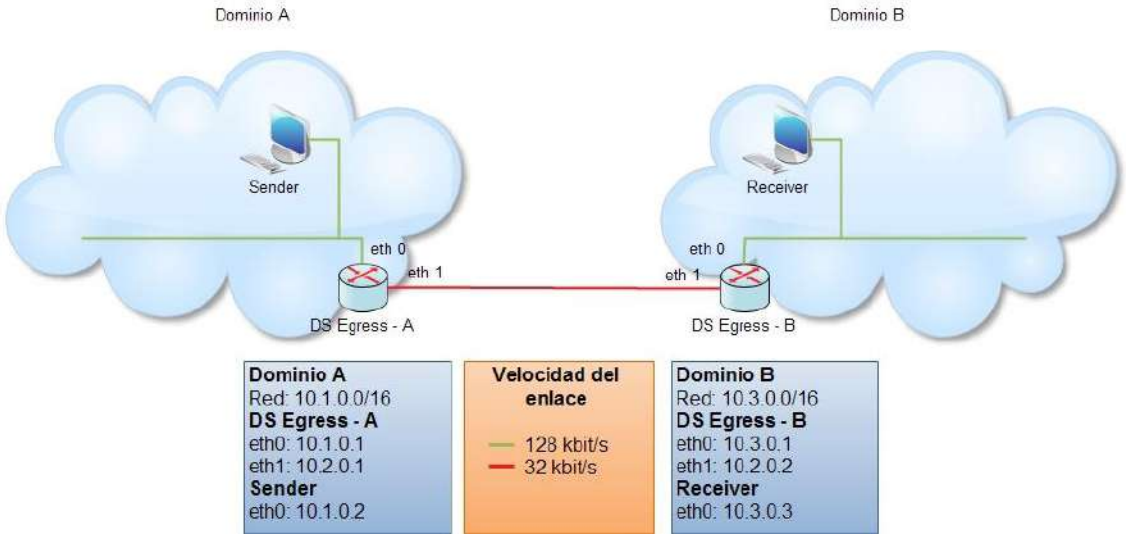


Figura 6.3.1: Escenario de prueba de ECN.

6.3.1. Herramientas

- **Tc:** Conjunto de mecanismos y operaciones por los cuales los paquetes son encolados en una interface de red para su trasmisión o recepción.
- **Tcpdump:** Herramienta que permite capturar y mostrar en tiempo real los paquetes que son transmitidos por la red.
- **Tcptrace:** Herramienta especializada en analizar el tráfico TCP.
- **JTG:** Generador de tráfico de red que permite enviar paquetes de características específicas que permiten emular flujos reales de red.

6.3.2. Casos de prueba

Se realizaron numerosas pruebas transmitiendo segmentos TCP de 1000 bytes de longitud durante 2 minutos. Se observó cómo se comportó la red utilizando las disciplinas Token Backet Filter (droptail), RED sin ECN y RED con ECN bajo los algoritmos de control de congestión Cubic y Reno. En estas experiencias se analizaron y compararon los parámetros de evaluación seleccionados y descriptos en la Tabla 6.3.1.

Parámetros de evaluación	
Duración de la transmisión	Tiempo transcurrido desde el inicio de la transmisión hasta el final de la conexión.
Tamaño de los paquetes	Tamaño máximo del segmento.
Paquetes transmitidos	Cantidad total de paquetes analizados en la captura.
Paquetes retransmitidos	Cantidad total de paquetes detectados como retransmisiones en la captura.
Máxima cantidad de retransmisiones	Cantidad máxima de retransmisiones encontradas en algún segmento.
Throughput (Bps)	Promedio entre la cantidad de bytes enviados únicos dividido la duración de transmisión.
Min owin (bytes)	Cantidad mínina de datos transmitidos sin confirmar en bytes distinto de cero.
Max owin (bytes)	Cantidad máxima de datos transmitidos sin confirmar en bytes observado en algún momento durante la transmisión.
Avh owin (bytes)	Cantidad promedio de datos transmitidos sin confirmar en bytes, calculados como la suma total de bytes salientes de las muestras dividido el total de las muestras.
Máximo tiempo ocioso	Máximo tiempo transcurrido entre el envío de dos paquetes consecutivos cualquiera.
RTT Promedio	Promedio de valores RTT (Round trip time) calculados como la suma total de RTT encontrados dividido el total de las muestras.

Tabla 6.3.1: Parámetros de evaluación.

La característica principal de Cubic es que la función de crecimiento de la ventana de congestión está definida en tiempo real y dicho crecimiento es independiente del RTT de los flujos. Mejora la escalabilidad de TCP sobre redes de alta velocidad y grandes latencias. Además, alcanza asignaciones equilibradas de ancho de banda a gran cantidad de flujos con distintos RTTs. La función de crecimiento está dada por una función cúbica que hace que durante etapas estables el tamaño de la ventana se incremente agresivamente mientras que dicha ventana se encuentre lejos de un punto de saturación. Mientras que se incrementará lentamente cuando se encuentre cerca a este punto. En consecuencia, intenta lograr encontrar un máximo en cuál mantener la ventana durante un período de tiempo prolongado.

Reno induce pérdidas de paquetes para estimar el ancho de banda disponible en la red. Mientras no se producen pérdidas Reno incrementa la ventana de congestión en 1 por cada RTT. Cuando experimenta pérdidas de paquetes, reduce la ventana de congestión a la mitad del tamaño actual. A esto se lo llama *incremento aditivo* y *decremento multiplicativo*. En consecuencia, Reno provoca un oscilamiento periódico en el tamaño de la ventana debido a la constante actualización del tamaño de la misma. Por lo tanto, las conexiones que presentan retrasos más cortos se ven favorecidas debido a que actualizan el tamaño de la ventana más rápidamente en comparación a conexiones de mayor retraso, resultando en una distribución injusta del ancho de banda entre los flujos de mayor RTT con los de menor RTT.

6.3.1. Configuraciones

Las configuraciones que fueron necesarias para llevar a cabo cada una de las pruebas antes mencionadas se pueden encontrar en el Apéndice C. A continuación daremos una explicación de las mismas.

6.3.1.1. Disciplina Token Backet Filter (Drop Tail)

Para llevar a cabo la experiencia utilizando Drop Tail, en el Router 1 se establece una velocidad de envío de 128kbit/s con ráfagas de hasta 8 paquetes y un buffer de 100 paquetes para la interfaz ETH 0. Para la interfaz ETH 1, se establece una velocidad de envío de 32kbit/s con ráfagas de hasta 2 paquetes y un buffer de 100 paquetes. El Router 2 presenta una configuración asimétrica con respecto al anterior. De esta forma se facilita la creación de un cuello de botella cuando ambos dominios se comuniquen entre sí, dando como resultado final una congestión en la red la cual es el objetivo de estudio.

6.3.1.2. Disciplina RED sin ECN

Debido a que la disciplina RED no es capaz de realizar “shaping” (limitar o distribuir el tráfico para lograr o no exceder una tasa configurada) fue necesario utilizar otra disciplina por encima de esta para lograr respetar las tasas de velocidades del caso anterior. En este caso, se utilizó la disciplina Hierarchical Token Bucket, que tal como se explicó capítulos anteriores es una disciplina con clases capaz de realizar la acción deseada, y a su vez, es capaz de combinar distintas disciplinas para lograr una organización de paquetes más refinada. Por este motivo, la configuración del Router 1

presenta una disciplina raíz HTB para lograr la potencia de varias disciplinas, una clase intermedia también HTB encargada de mantener la misma velocidad de transmisión utilizada con TBF, y una clase hoja RED encargada de marcar los paquetes para su descarte de acuerdo a dos umbrales previamente definidos y a una probabilidad de descarte de acuerdo al ancho de banda que demanden los flujos a los que pertenezcan. El objetivo de estos umbrales es evitar descartar los paquetes por el desbordamiento del tamaño del buffer de memoria del router y el objetivo de la probabilidad de descarte es brindar un tratamiento más justo a los distintos flujos que en ese instante están atravesando el router. Nuevamente, el Router 2 presenta una configuración asimétrica al Router 1 para facilitar la creación de un cuello de botella.

6.3.1.3. Disciplina RED con ECN

Las configuraciones son exactamente iguales a las presentadas en el caso anterior con la única diferencia de la aparición del parámetro ***ecn*** para indicar la utilización de la notificación de congestión explicita. Este parámetro le indica a la disciplina que debe marcar el paquete en el header IP en lugar de descartarlo ya sea por haber sido seleccionado por la función de probabilidad o porque el tamaño del buffer supera el umbral máximo definido, como sería en su funcionamiento convencional.

6.3.2. Mecanismo ECN en marcha

Antes de pasar al resultado de las pruebas realizadas, en este apartado mostraremos como funciona ECN en la práctica sobre el protocolo TCP desde el comienzo de la conexión. Esto será posible mediante el análisis de una de las capturas que fueron realizadas para este trabajo.

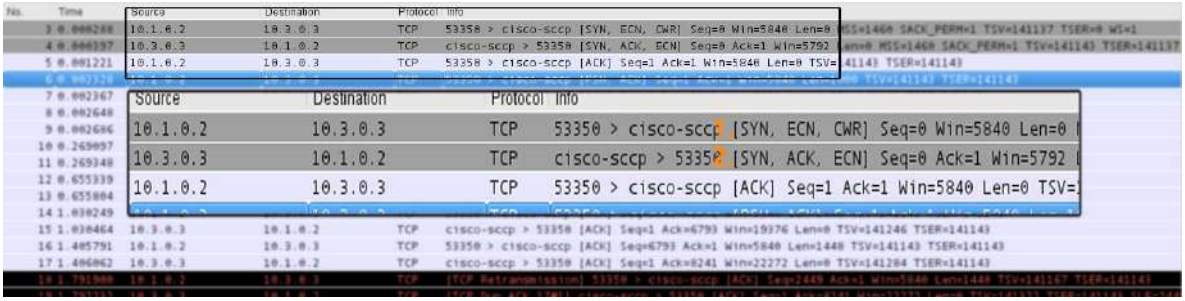


Figura 6.3.2: Acuerdo de 3 vías ECN.

En primer lugar, podemos observar en la Figura 6.3.2 cómo se lleva a cabo el acuerdo de 3 vías tradicional del protocolo TCP acompañado de las banderas que indican la compatibilidad con

notificación de congestión explícita. Marcamos con 1 sobre la captura al correspondiente paquete SYN ECN-Setup y con 2 al paquete SYN-ACK ECN-Setup mencionados en el 0.

En detalle, podemos observar las banderas sobre estos paquetes:

1. SYN ECN-Setup.

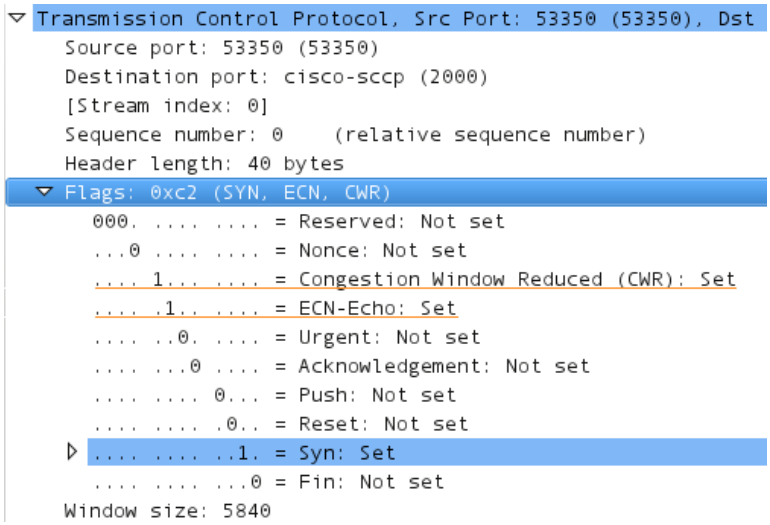


Figura 6.3.3: Paquete SYN ECN-Setup detallado.

2. SYN-ACK ECN-Setup.

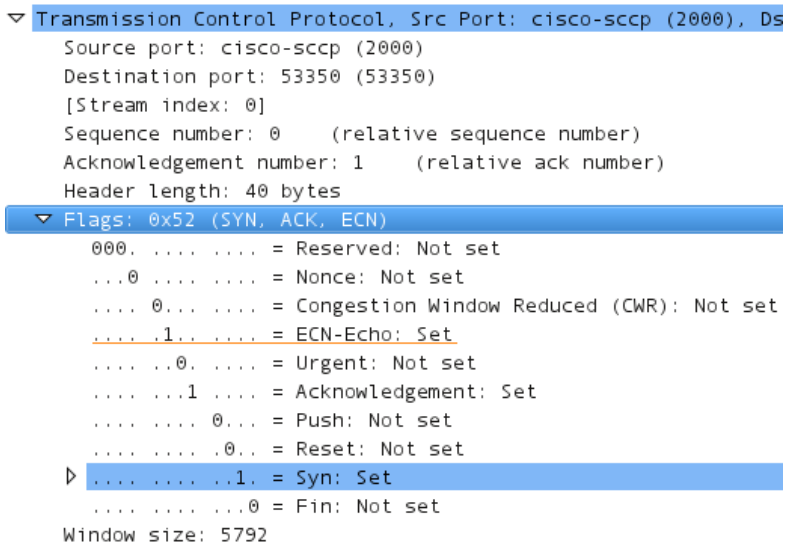


Figura 6.3.4: Paquete SYN-ACK ECN-Setup detallado.

Una vez establecida la conexión entre los extremos, estos comienzan a transmitir información normalmente hasta producirse la congestión.

En la siguiente captura se muestra el momento en el que el Receptor detecta la congestión producto de haber recibido por parte del router Egress-B (originalmente enviado desde el router Egress-A) un paquete CE de notificación de congestión a nivel de capa de red IP, el siguiente ACK que envía el Receptor de los datos para dicho paquete va acompañado de la bandera ECN que notifica la congestión a nivel de capa de transporte TCP.

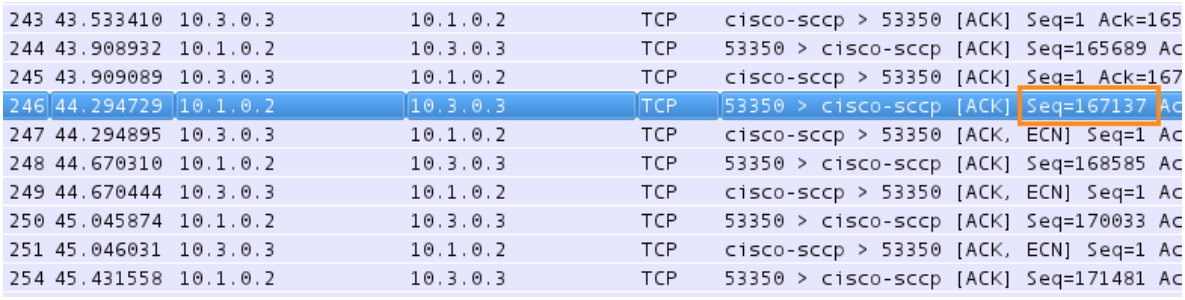


Figura 6.3.5: Detección de congestión a nivel IP.

Esta notificación de congestión es transmitida hasta que el Emisor anuncie que ha reducido su ventana de transmisión de datos y de esta manera evitar o reducir la congestión.

El paquete marcado en la captura anterior es el siguiente:

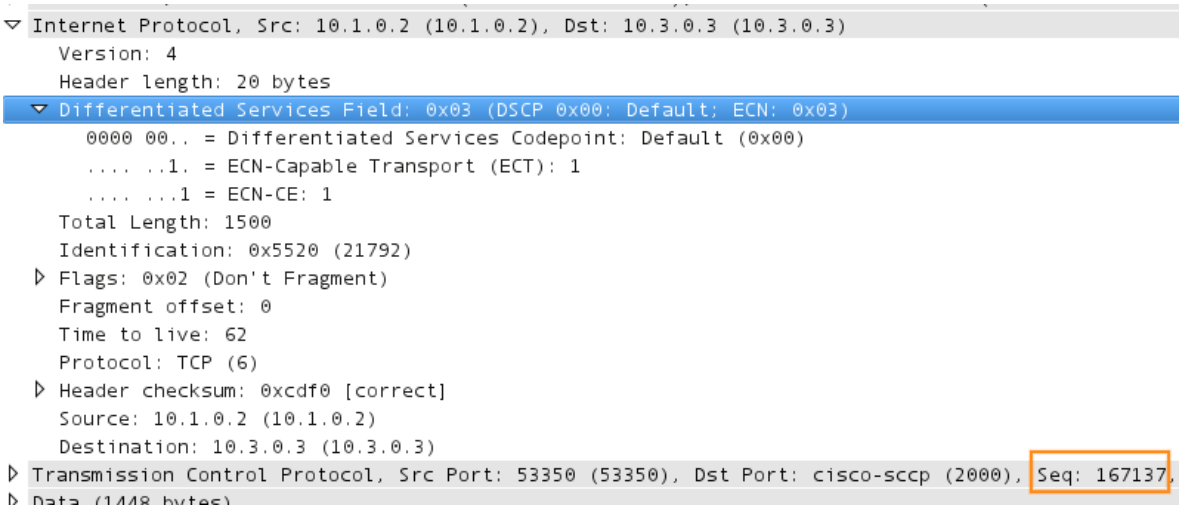


Figura 6.3.6: Paquete CE detallado.

La notificación de congestión a nivel IP, proveniente de los routers Egress, es marcada en los últimos dos bits del campo DSCP como fue definido en RFC 2481.

El Emisor recibe una ráfaga de paquetes con la notificación de congestión antes que pueda responder. Este no responde ante el primer paquete debido a que la ráfaga es consecuencia de la congestión provocada por el cuello de botella que se produce en los routers extremos.

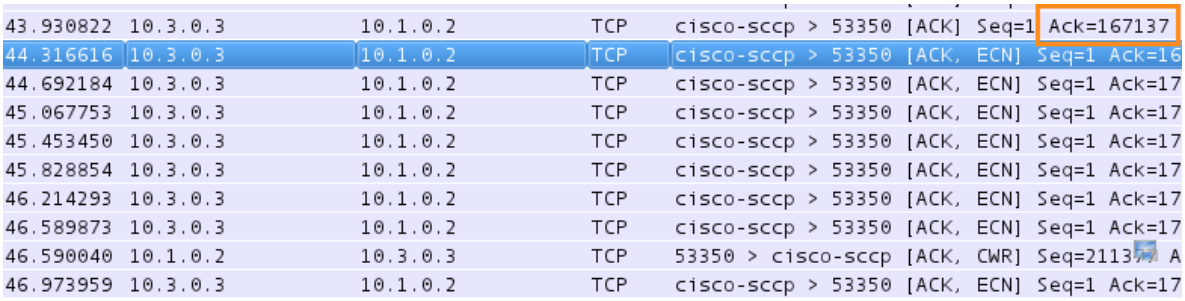


Figura 6.3.7: Recepción y respuesta a la notificación ECN.

El Receptor seguirá enviando la bandera ECN no obstante reciba la bandera CWR por parte del Emisor, y éste a su vez no seguirá respondiendo con esta bandera ante los sucesivos paquetes ECN ya que el mecanismo de control de congestión explícita responde una sola vez por RTT.

El Receptor de los datos recibe la notificación CWR del Emisor y ante esta situación comienza a transmitir los ACK sin la bandera ECN como sucediera antes de detectarse la congestión en los enlaces.

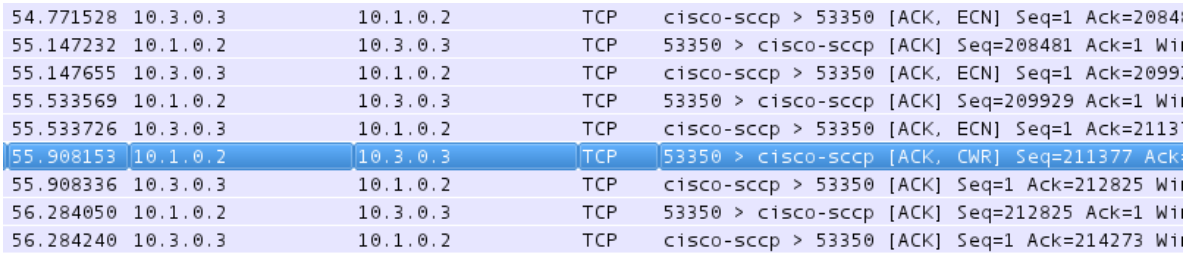


Figura 6.3.8: Recepción de notificación CWR.

El tiempo transcurrido entre el envío del primer paquete con la bandera ECN y la recepción del paquete con la bandera CWR que le corresponde consta de aproximadamente 12 segundos para el Receptor, cuando en realidad el Emisor tardó solamente en responder aproximadamente 3 segundos. Esto significa que el tiempo estimado de retardo por congestión en los enlaces

correspondió a 9 segundos. Esta conclusión puede corroborarse fácilmente observando los tiempos del paquete con la bandera CWR. El envío se produjo aproximadamente a los 47 segundos de iniciada la conexión mientras que la recepción de dicho paquete fue a los 56 segundos de la misma, dando una diferencia estimada de 9 segundos.

Para tener más claridad sobre este hecho podemos observar las siguientes líneas de tiempo que corresponden a la conexión anterior, donde los ejes vertical y horizontal son representados por el número de secuencia de los paquetes y la duración de la conexión respectivamente:

Desde el Receptor podemos ver el momento exacto en el que envía el primer ECN, tras recibir la notificación de congestión sobre el protocolo IP – bit CE -, y el momento en que recibe la respuesta del Emisor, tal como lo muestra la Figura 6.3.9. Además se observan congestiones producidas posteriormente durante la misma conexión.

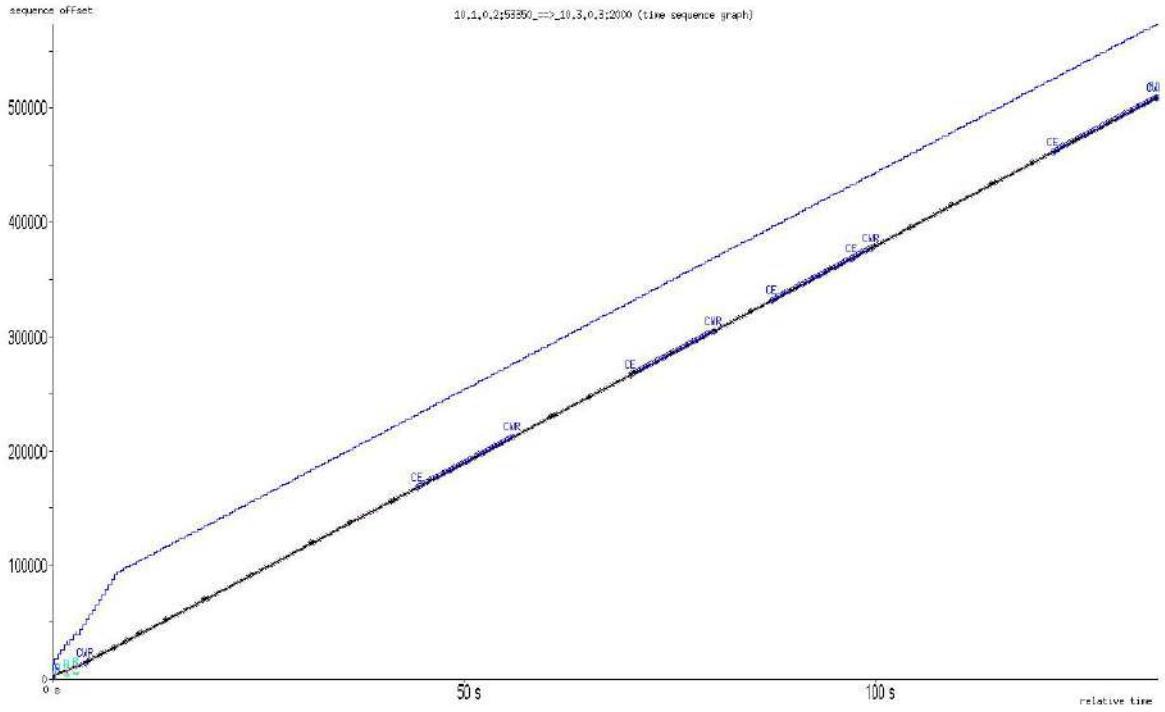


Figura 6.3.9: Línea de tiempo de la conexión vista en el Receptor.

La línea azul representa la ventana de recepción anunciada desde el otro extremo de la conexión, en este caso del Emisor. Se dibuja a la altura del valor de secuencia correspondiente a la suma del número ACK actual y la ventana de recepción del último paquete ACK recibido. Las flechas

verticales negras, que en el gráfico se aprecia como una línea negra continua, representan los segmentos enviados. La altura de la flecha está dada por los números de secuencia del primer y último byte de dicho segmento.

En una imagen ampliada (Figura 6.3.10) puede observarse aproximadamente los tiempos en que se genera la notificación de congestión y se recibe la notificación de reducción de la ventana de transmisión. La línea violeta representa los valores ACK recibidos desde el otro extremo de la conexión. Del mismo modo, en la Figura 6.3.11 se puede observar el mismo hecho desde el Emisor.

Puede notarse que el Emisor permaneció por algunos instantes de tiempo en estado ocioso, producto de la congestión y el agotamiento de su ventana de transmisión. Los puntos azules sobre la línea violeta corresponden a los paquetes recibidos con la bandera ECN. En la Figura 6.3.9 y la Figura 6.3.11 se observa flechas verticales de color celeste que representan segmentos retransmitidos de manera similar que se representan a los segmentos originalmente transmitidos.

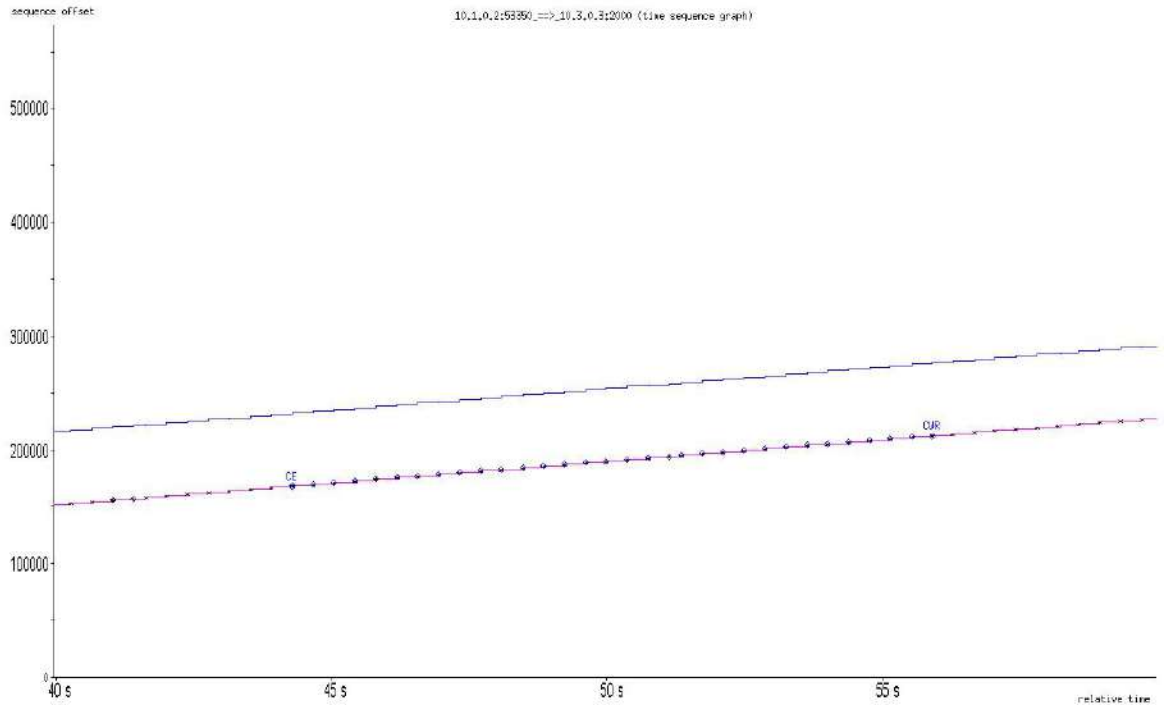


Figura 6.3.10: Notificaciones ECN durante la conexión (Receptor).

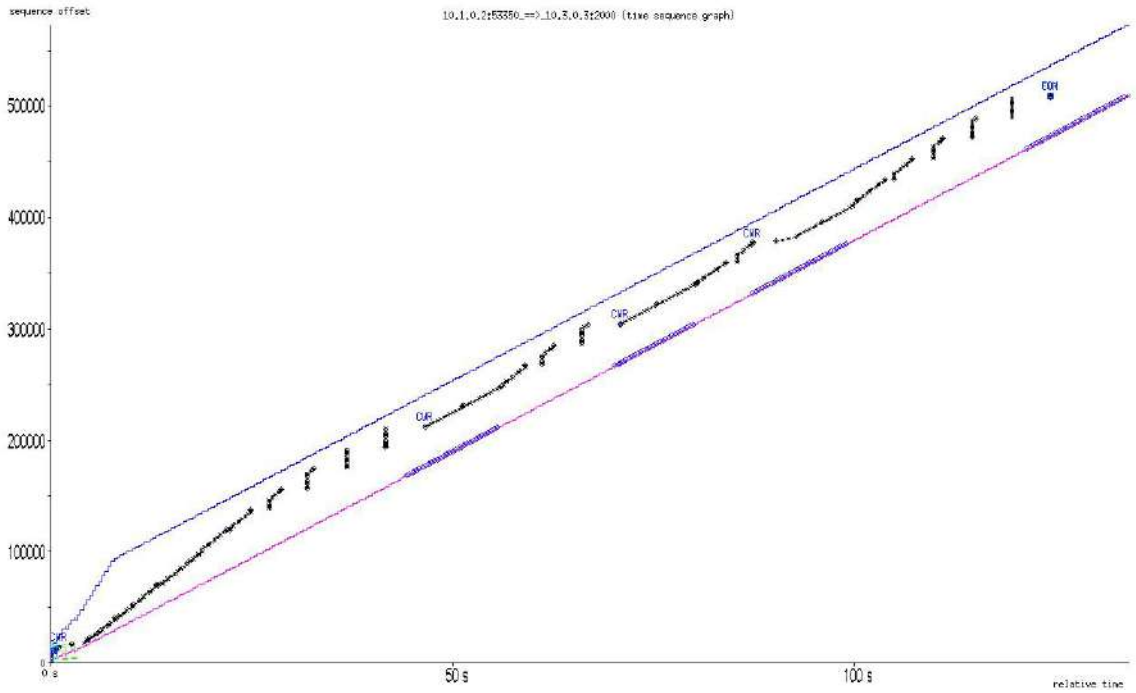


Figura 6.3.11: Línea de tiempo de la conexión vista en el Emisor.

La Figura 6.3.12 muestra desde el Emisor las notificaciones que son recibidas y enviadas.

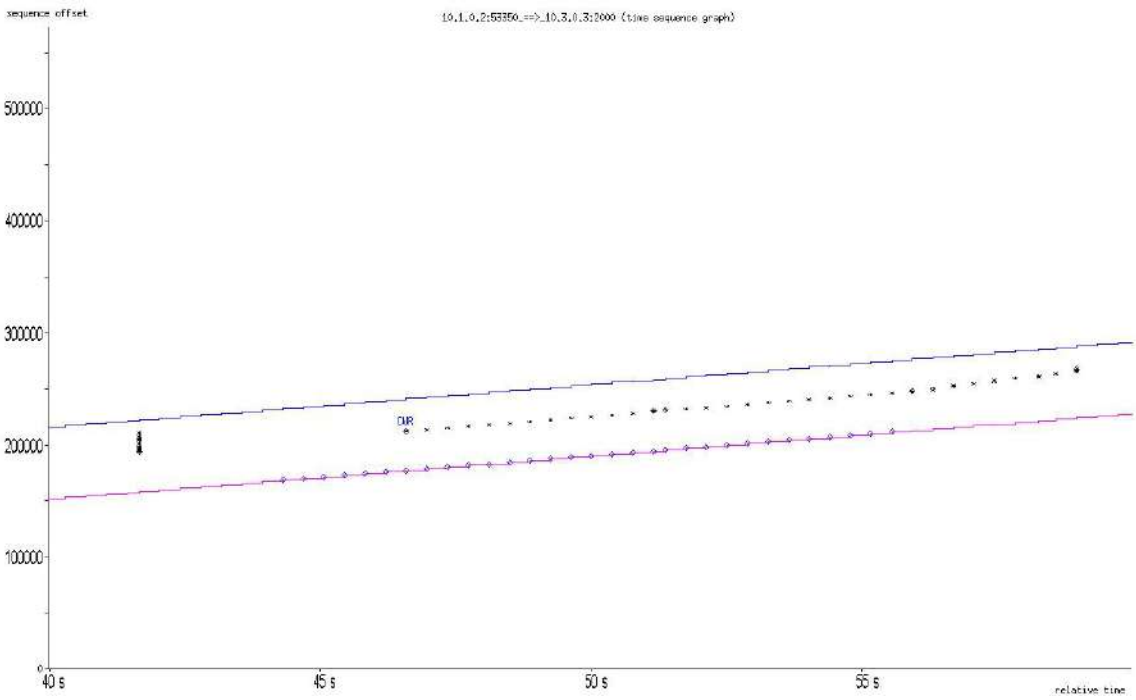


Figura 6.3.12: Notificaciones ECN durante la conexión (Emisor).

6.3.3. Pruebas y análisis de resultados

La intención es demostrar, a través de las pruebas, en primer lugar que el control de congestión explicito (ECN) mejora sensiblemente la utilización del ancho de banda en los enlaces y la comunicación entre los extremos disminuyendo latencia y retransmisiones por pérdidas de paquetes entre otras cosas. Y por otro lado, que ECN funciona independientemente del algoritmo de control de congestión que se esté utilizando sobre la capa de transporte, sin afectar significativamente sus características y los fines para los cuales fueron implementados.

La Tabla 6.3.2 y la Tabla 6.3.4 representan la media de los resultados obtenidos de las distintas pruebas realizadas, mientras que la Tabla 6.3.3 y la Tabla 6.3.5 corresponden al desvío estándar de los mismos en el orden respectivo. Las tablas detalladas con los valores de cada una de las pruebas realizadas pueden encontrarse en el Apéndice D.

	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	02:18,47	02:12,28	02:12,11
Tamaño de los paquetes (Kbytes)	1000	1000	1000
Paquetes transmitidos	667	716	716
Paquetes retransmitidos	36	7	1
Segmentos máximos retransmitidos	1	1	0
Throughput (Bps)	3.357	3.769	3.814
Min owin (bytes)	1	1	1
Max owin (bytes)	52.972	52.083	51.771
Avg owin (bytes)	33.426	27.029	34.291
Máximo tiempo ocioso (ms)	19.101,0	11.935,6	9.139,0
RTT Promedio (ms)	8.126,2	5.441,7	9.080,3

Tabla 6.3.2: Resultados con el algoritmo CUBIC.

	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	00:10,26	00:06,46	00:01,25
Tamaño de los paquetes (Kbytes)	0	0	0
Paquetes transmitidos	51	19	6
Paquetes retransmitidos	13	2	1
Segmentos máximos retransmitidos	0	0	1
Throughput (Bps)	429	117	11
Min owin (bytes)	0	0	0
Max owin (bytes)	37	1.492	1.142

Avg owin (bytes)	1.949	1.670	1.647
Máximo tiempo ocioso (ms)	9.666,4	4.346,8	729,8
RTT Promedio (ms)	701,9	536,6	386,4

Tabla 6.3.3: Desvío estándar de resultados con el algoritmo CUBIC.

	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	02:07,98	02:08,53	02:11,63
Tamaño de los paquetes (Kbytes)	1000	1000	1000
Paquetes transmitidos	687	693	707
Paquetes retransmitidos	9	3	1
Segmentos máximos retransmitidos	1	1	0
Throughput (Bps)	3.776	3.818	3.819
Min owin (bytes)	1	1	1
Max owin (bytes)	30.409	35.718	25.582
Avg owin (bytes)	16.043	16.782	17.431
Máximo tiempo ocioso (ms)	4.738,1	5.305,1	5.315,9
RTT Promedio (ms)	3.989,4	4.121,2	4.602,7

Tabla 6.3.4: Resultados con el algoritmo RENO.

	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	00:02,50	00:01,98	00:01,44
Tamaño de los paquetes (Kbytes)	0	0	0
Paquetes transmitidos	31	10	9
Paquetes retransmitidos	10	1	1
Segmentos máximos retransmitidos	0	0	0
Throughput (Bps)	207	10	7
Min owin (bytes)	0	0	0
Max owin (bytes)	7.859	2.851	1.718
Avg owin (bytes)	4.228	782	999
Máximo tiempo ocioso (ms)	7.591,4	883,8	796,2
RTT Promedio (ms)	1.051,9	202,0	251,9

Tabla 6.3.5: Desvío estándar de resultados con el algoritmo RENO.

Cabe destacar que el tiempo en las pruebas excede los dos minutos debido al cuello de botella que se produjo entre los routers. Si bien el Emisor transmite paquetes durante dos minutos, la totalidad de ellos tardan más en arribar. Observando los valores expuestos en las tablas anteriores podemos arribar a los siguientes resultados:

- La duración de transmisión fue menor utilizando Reno que utilizando Cubic. Pero al utilizar ECN la diferencia entre ambos algoritmos no fue tan grande.

- Siguiendo con el tiempo de duración de las conexiones, ECN obtuvo menor duración utilizando Cubic y mayor duración utilizando Reno. Siendo curioso que la menor duración con Reno fue utilizando la disciplina Droptail, que a su vez obtuvo la mayor duración con Cubic.

- ECN genera la menor cantidad de retransmisiones y la mayor cantidad de transmisiones de paquetes, tanto en Cubic como en Reno. Por otro lado, Droptail fue la disciplina que mayor cantidad de retransmisiones realizó y la que menor cantidad de paquetes transmitió.

- De los puntos anteriores sobresale que las disciplinas Droptail y RED con ECN siempre tuvieron desempeños opuestos y la utilización de RED sin ECN tiene un desempeño intermedio a las anteriores.

- RED con ECN fue la disciplina más estable, ya que el tiempo de transmisión entre las pruebas fue de 1 segundo aproximadamente (tanto para Cubic como con Reno), además es la que menor variación del throughput y cantidad de paquetes retransmitidos presenta.

- La ventana de congestión promedio siempre fue mayor utilizando ECN. Siendo además siempre mayor utilizando Cubic que Reno, esto se debe a las características propias de cada algoritmo.

- Una curiosidad fue observar que el mayor RTT se produjo utilizando ECN. La topología, la configuración de los host y de las disciplinas no varían significativamente, sin embargo esto puede llegar a ser consecuencia del procesamiento adicional que deben realizar los extremos y los routers sobre los paquetes.

Como conclusión, puede observarse la obtención de resultados esperados según la teoría desarrollada en los capítulos anteriores. ECN ha sido la disciplina con mayor cantidad de paquetes transmitidos, menor retransmisión y mayor rendimiento, tanto en CUBIC como en RENO. En contraposición, Droptail ha sido el eslabón más débil en estos términos. Es decir, ECN ha sido independiente del algoritmo de control de congestión implementado por TCP, y su utilización significo una mejora en la comunicación.

Capítulo 7: Casos Prácticos de Alta Disponibilidad

7.1. Introducción

El objetivo de este capítulo es exponer los conceptos de alta disponibilidad planteados en los capítulos anteriores para la puesta en marcha del clúster de telefonía IP capaz de soportar una gran cantidad de usuarios en forma concurrente.

Los casos prácticos elaborados en este capítulo fueron realizados en equipos reales. Esto se debió a que la complejidad del escenario, sumado a la cantidad de entidades intervinientes y a los servicios implementados, dificulta enormemente su elaboración en máquinas virtuales. Además, en caso de haber querido realizarlo sobre una arquitectura UML hubiera implicado la necesidad de poseer una computadora lo suficientemente potente para correr dos balanceadores de carga, dos servidores de registro, dos servidores de lookup, un Gateway PSTN, y varios clientes.

7.2. Topología

La topología utilizada para la realización de las pruebas de alta disponibilidad se muestra en la Figura 7.2.1. La misma está compuesta por dos hosts balanceadores de carga, dos servidores de registro, dos de lookup (Asterisk y bases de datos) y un Gateway PSTN, además de los clientes que hacen uso de los servicios ofrecidos por el clúster.

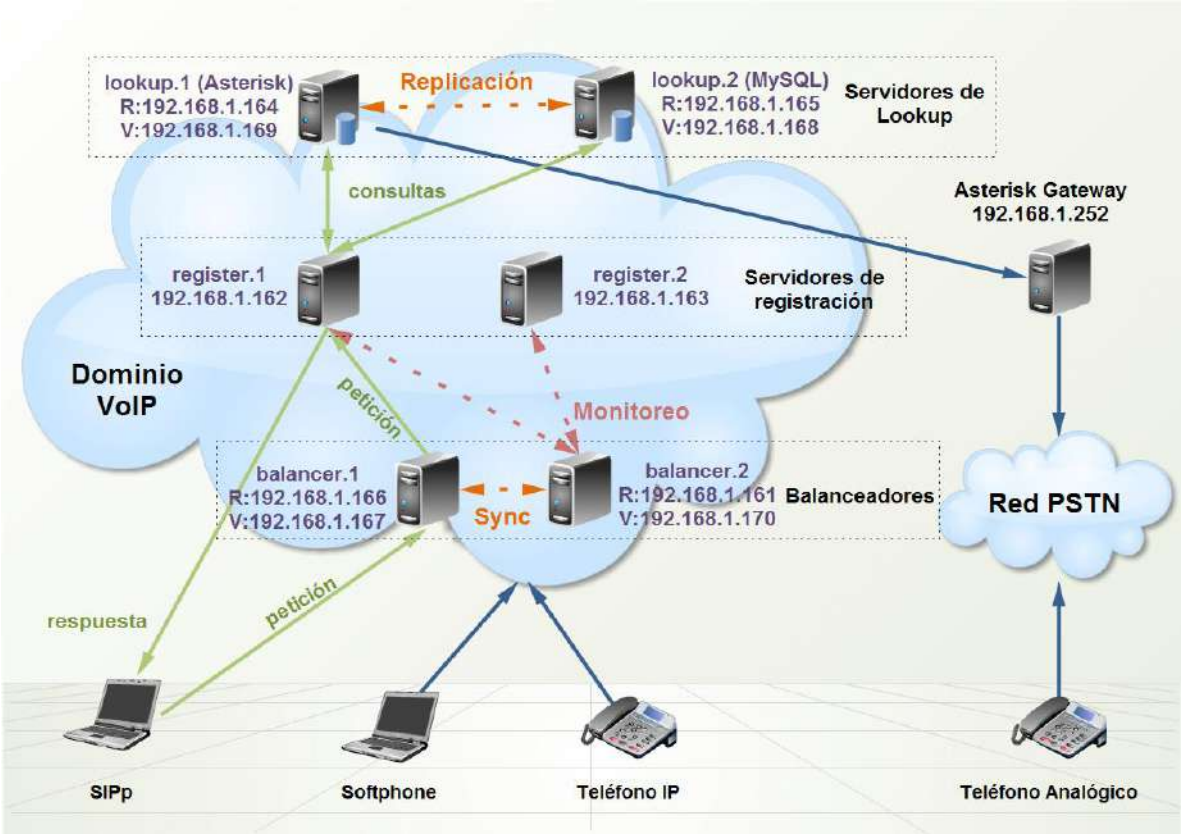


Figura 7.2.1: Topología Clúster.

7.2.1. Características del Hardware

Las características técnicas del hardware utilizado para el escenario son las que se detallan a continuación:

- Servidores de lookup y bases de datos:

Nombre	Procesador	Memoria	Disco rígido	Placa de red	Dirección IP
lookup.1	Pentium Dual Core	2Gb	300Gb	100Mbps	192.168.1.164
lookup.2	Pentium Dual Core	2Gb	300Gb	100Mbps	192.168.1.165

- Servidores de registro:

Nombre	Procesador	Memoria	Disco rígido	Placa de red	Dirección IP
register.1	Intel i3	4Gb	500Gb	100Mbps	192.168.1.162
register.2	Intel i3	4Gb	500Gb	100Mbps	192.168.1.163

- Gateway PSTN:

Nombre	Procesador	Memoria	Disco rígido	Placa de red	Dirección IP
gateway	Pentium Dual Core	2Gb	160Gb	100Mbps	192.168.1.252

- Balanceadores de carga:

Nombre	Procesador	Memoria	Disco rígido	Placa de red	Dirección IP
balancer.1	Pentium Dual Core	2Gb	300Gb	100Mbps	192.168.1.166
balancer.2	Pentium Dual Core 1.86GHz	2Gb	160Gb	100Mbps	192.168.1.161

7.3. Configuraciones

7.3.1. Balanceadores de carga

El balanceo de carga se realiza a través de dos nodos balanceadores que funcionan en modo activo-activo con IPVS y en alta disponibilidad mediante Heartbeat. Para lograr la funcionalidad activo-activo se utilizan dos direcciones IP “públicas” a través de las cuales acceden los clientes. Estas direcciones son balanceadas entre los dos nodos con Heartbeat y ante el desperfecto de alguno de ellos, la misma es asignada al nodo par. Por otro lado, el software de balanceo (IPVS) está configurado para funcionar en modo master-master permitiendo así que los nodos compartan entre ellos las conexiones activas. Esto se consiguió a través de dos configuraciones master-slave invertidas en ambos servidores.

La función de los balanceadores es realizar una distribución equitativa de todas las conexiones que reciben por parte de los clientes direccionándolas hacia los servidores reales. Es importante destacar que éstos no balancean servicios, sino que balancean todas las conexiones sin importar el puerto destino. Para ello, se utiliza un tipo de configuración denominado Fmark, que consiste en marcar los paquetes entrantes y re-direccionar todos aquellos que tengan la misma marca hacia un mismo destino. Para lograr el tipo de balanceo equitativo mencionado se utilizó una estrategia de balanceo WLC (Weighted Least Connection), que consiste en enviar las conexiones entrantes al servidor con menor cantidad de conexiones asignadas y además teniendo en consideración el peso asignado a cada servidor. En nuestro caso, dado que las características de hardware de los servidores reales son iguales, se determinó utilizar el mismo peso para ambos.

La implementación de los balanceadores se realizó mediante Ldirectord. Esta herramienta se encarga de la sincronización de los nodos y del monitoreo de los servicios que se ejecutan en los servidores de registro. Su configuración se encuentra en el archivo *ldirector.cf* y se muestra a continuación:

```
checktimeout=3
checkinterval=3
logfile="/var/log/ldirectord.log"
virtual = 123
        real=192.168.1.162 ipip 1
        real=192.168.1.163 ipip 1
        checkcommand="/usr/local/bin/asterisk_monitor.sh"
        scheduler="wlc"
        protocol=fwm
        checktype=external
        checktimeout=15
```

El monitoreo de los servidores de registro se realiza a través del script "`/usr/local/bin/asterisk_monitor.sh`" el cual utiliza la herramienta de testeo SIPp, que se detallará más adelante.

El marcado de paquetes para la estrategia Fmark se llevó a cabo utilizando Iptables, las reglas que lograron esto son:

```
iptables -t mangle -A PREROUTING -d 192.168.1.167 -j MARK --set-mark 123
iptables -t mangle -A PREROUTING -d 192.168.1.170 -j MARK --set-mark 123
```

Como se puede observar las direcciones IP a través de las cuales se recibe el tráfico son 192.168.1.167 y 192.168.1.170. Para lograr la alta disponibilidad de los servicios del clúster se utilizó Pacemaker y se lo configuró de la siguiente manera:

```
node $id="0afb23a9-6f17-4f2e-8cc3-fe071247d08d" balancer.2
node $id="8645f1e5-f0bb-4cfb-a620-17623d0bbc49" balancer.1
primitive BalanceIP1 ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.170" cidr_netmask="32" lvs_support="true" \
       op monitor interval="15s"
primitive BalanceIP2 ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.167" cidr_netmask="32" lvs_support="true" \
       op monitor interval="15s"
primitive lDirector ocf:heartbeat:ldirectord \
       paramsconfigfile="/etc/ldirectord.cf" \
       ldirectord="/usr/local/sbin/ldirectord"
clone CL_ldirector lDirector \
       meta clone-max="2" clone-node-max="1" globally-unique="false" \
       target-role="Started"
location ip1-balancer.1 BalanceIP1 250: balancer.1
location ip2-balancer.2 BalanceIP2 250: balancer.2
order ip-after-ldirector1 inf: CL_ldirector BalanceIP1
order ip-after-ldirector2 inf: CL_ldirector BalanceIP2
property $id="cib-bootstrap-options" \
       expected-quorum-votes="2" \
       no-quorum-policy="ignore" \
       stonith-enabled="false" \
       dc-version="1.1.6-ad7c08735dfe3e633777a512b1f9cfcb584dcf1b" \
       cluster-infrastructure="Heartbeat"
rsc_defaults $id="rsc-options" \
       resource-stickiness="100"
```

En esta configuración se puede observar las primitivas *BalanceIP1ocf:heartbeat:IPaddr2 y BalanceIP2 ocf:heartbeat:IPaddr2* que referencian a las dos direcciones IP que son balanceadas. La primitiva *ocf:heartbeat:ldirectord* que especifica el uso del servicio Ldirectord. La directiva *clone* que especifica que el servicio Ldirectord debe ser ejecutado en los dos servidores de balanceo. Las directivas *location* sirven para darle prioridad a los servidores en la asignación de las direcciones IP. Finalmente, la primitiva *order* sirve para asegurar que la IP flotante es asignada al servidor solo después de haber sido ejecutado el servicio Ldirectord. Esto se hace para evitar indisponibilidad en el servicio.

La idea de utilizar dos direcciones IP públicas se basa en que ambos balanceadores puedan ser accedidos de forma transparente a través de un único nombre de dominio que resuelva a éstas direcciones IP. Es decir, realizando un balanceo de carga a nivel DNS. Esto no fue implementado en

el proyecto debido a que es ajeno al alcance de los objetivos planteados.

7.3.2. Servidores de lookup y bases de datos

La función de estos servidores es la de intermediar en la búsqueda de usuarios dentro del sistema, a través de la distribución de las solicitudes de lookup del protocolo DUNDi. Además, cuenta con una base de datos centralizada MySQL configurada con una replicación master-master donde los servidores de registro pueden consultar los usuarios configurados y habilitados, las extensiones de llamada y cuentas de mail de correo de voz, es decir, todos los datos que deben estar disponibles y ser independientes de los servidores de registro para lograr escalabilidad en el sistema. Dado que estos servicios son críticos para el funcionamiento, se encuentran funcionando en ambos servidores y presentan una arquitectura activo-pasivo administrado a través de Heartbeat. Con el fin de lograr una mejor utilización de los recursos de hardware, el servidor lookup.1 funciona principalmente como servidor de lookup y el servidor lookup.2 funciona como servidor de base de datos, pero ante la falla de alguno de estos, el servicio es migrado al otro servidor. Para lograr esto, Heartbeat realiza el balanceo de dos direcciones IP flotantes, una para cada servicio. A continuación se muestra la configuración de Pacemaker que logra este objetivo:

```
node $id="1392955f-c953-40ff-98b3-d9264e5ab458" lookup.2
node $id="b39a3b63-ab82-4a54-8d68-64c6f8c4b261" lookup.1
primitive Asterisk lsb:asterisk \
       op monitor interval="20s"
primitive ClusterIP ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.169" cidr_netmask="32" \
       op monitor interval="15s"
primitive MySQL ocf:heartbeat:mysql \
       params binary="/usr/bin/mysqld_safe" datadir="/var/lib/mysql"
pid="/var/run/mysqld/mysqld.pid" user="mysql" log="/var/log/mysqld.log"
socket="/var/lib/mysql/mysql.sock" test_table="asterisk.sipfriends"
test_user="test" test_passwd="testpassword" \
       op start interval="0" timeout="120" \
       op monitor interval="20" timeout="60" depth="0" \
       op stop interval="0" timeout="120"
primitive MySQL_IP ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.168" cidr_netmask="32" \
       op monitor interval="15s"
clone CL_MySQL MySQL \
       meta clone-max="2" clone-node-max="1" globally-unique="false"
location mysql-prefer-lookup.2 CL_MySQL 250: lookup.2
location mysqlip-prefer-lookup.2 MySQL_IP 250: lookup.2
location prefer-lookup.1 Asterisk 250: lookup.1
colocation asterisk-with-ip inf: Asterisk ClusterIP
colocation master1-with-ip inf: MySQL_IP CL_MySQL
order asterisk-after-ip inf: ClusterIP Asterisk
order ip-after-mysql inf: CL_MySQL MySQL_IP
property $id="cib-bootstrap-options" \
       dc-version="1.1.6-ad7c08735dfe3e633777a512b1f9cfcb584dcf1b" \
       cluster-infrastructure="Heartbeat" \
       stonith-enabled="false" \
       expected-quorum-votes="2" \
       no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
       resource-stickiness="100"
```

De forma similar a como se explicó anteriormente, en este caso se configuraron cuatro primitivas. La primitiva *lsb:asterisk* que corresponde al servicio Asterisk interviniente en las búsquedas de usuarios a través de DUNDi, la primitiva *ocf:heartbeat:mysql* correspondiente al servicio MySQL y las dos primitivas *ocf:heartbeat:IPaddr2* correspondientes a las dos direcciones IP flotantes (192.168.1.168 y 192.168.1.169) una para cada servicio. Dado a que MySQL funciona como master-master, el servicio debe estar funcionando en los dos servidores, esto se logra con la directiva *clone*.

La directiva *colocation* especificada en la configuración es la que se encarga de asociar cada dirección IP a un determinado servicio. De esta forma, la dirección IP 192.168.1.168 siempre corresponde al servicio MySQL y la dirección IP 192.168.169 al Asterisk de lookup.

Por otro lado, la replicación master-master entre los servicios de MySQL se logra a través de una doble replicación master-slave. Para esto, fue necesario configurar dos usuarios MySQL con permisos *replication slave* en ambos servidores y agregar en los archivos de configuración de cada servidor, eurocase4 y eurocase5 respectivamente, lo siguiente:

```
[mysqld]
server-id=1
log-bin
binlog-do-db=asterisk
binlog-ignore-db=mysql
binlog-ignore-db=test
master-host=lookup.1
master-user=eurocase4
master-password=eurocase4
```

```
[mysqld]
server-id=2
log-bin
binlog-do-db=asterisk
binlog-ignore-db=mysql
binlog-ignore-db=test
master-host=lookup.2
master-user=eurocase5
master-password=eurocase5
```

Para crear los usuarios de replicación se utilizan las siguientes sentencias SQL en los servidores correspondientes:

```
grant replication slave on *.* to 'eurocase4'@lookup.1 identified by
'eurocase4';

grant replication slave on *.* to 'eurocase5'@lookup.2 identified by
'eurocase5';
```

Para brindar el servicio de lookup fue necesario configurar todos los nodos pares involucrados. Principalmente los servidores que centralizan las consultas (lookup.1 y lookup.2) debieron tener en su configuración los registros correspondientes a todos los servidores de registro del sistema. De esta forma se logra que los servidores de registro solo deban tener configurado un único par, simplificando notablemente la configuración y la escalabilidad del sistema, dado que al agregar un nuevo servidor al clúster solo debe configurarse en los servidores de lookup centralizados. A continuación se muestra el archivo de configuración *dundi.conf* de estos servidores:

```
[general]
department=Lab Redes
organization=UNLPam
locality=General Pico, La Pampa
stateprov=LP
country=AR
email=asterisk@gmail.com
phone=+12565551212
bindaddr=192.168.1.169
port=4520
entityid=6C:F0:49:A4:4B:98
cachetime=5
ttl=1
autokill=yes

[mappings]
priv =>

[B8:AC:6F:68:A9:01] ; regiter.1
model = symmetric
host = 192.168.1.162
inkey = dundi
outkey = dundi
include = priv
permit = priv
qualify = yes
order = primary

[B8:AC:6F:68:A9:70] ; register.2
model = symmetric
host = 192.168.1.163
inkey = dundi
outkey = dundi
include = priv
permit = priv
qualify = yes
order = primary
```

7.3.3. Servidores de registro

Estos servidores son los encargados de atender todas las solicitudes de los clientes, las cuales fueron re-direccionadas desde los servidores de balanceo de carga. Su configuración es relativamente sencilla y muy similar en cada uno de los servidores, lo que permite poder agregar o quitar servidores al clúster con muy poco esfuerzo. Estos servidores deben ser capaces de procesar paquetes con encapsulamiento IP a través del uso del módulo IPIP presente en el Kernel Linux. Esto se debe a que los paquetes recibidos fueron encapsulados previamente por el balanceador con el motivo de que los servidores de registro puedan obtener el paquete original enviado por el cliente. Esto permite responder de forma directa al cliente sin necesidad de volver a pasar por el balanceador, disminuyendo así el overhead del proceso y permitiendo al balanceador

mayor capacidad de procesamiento.

En este sentido los servidores de registro deben poder recibir paquetes enviados a las direcciones públicas de los balanceadores de carga, y por lo tanto deben tener configurados en sus interfaces de red estas direcciones IP. Esto se hace con los siguientes comandos:

```
ifconfig tunl0 192.168.1.170  netmask 255.255.255.255 broadcast 192.168.1.170 up
route add –host 192.168.1.170 dev tunl0
ifconfig tunl1 192.168.1.167  netmask 255.255.255.255 broadcast 192.168.1.167 up
route add –host 192.168.1.167 dev tunl1
```

Se debe notar que no se trata de interfaces de red reales sino de interfaces TUNL (Túneles IP), éstas son unas interfaces especiales utilizadas para el encapsulamiento IP.

Así como en los servidores de lookup y bases de datos se configuraron DUNDi y MySQL, en los servidores de registro se debe configurar a Asterisk para que haga uso de estos servicios. Por un lado, se debió especificar los parámetros de configuración para poder acceder a la base de datos. Esto se logró mediante la edición del archivo de configuración *res_config_mysql.conf* como se muestra a continuación:

```
[general]
dbhost = 192.168.1.168
dbname = asterisk
dbuser = asterisk
dbpass = eurocase@asterisk
dbport = 3306
requirements=warn
```

Además, se debió modificar el archivo *extconfig.conf* para especificar qué información obtener y desde qué tabla se obtiene, resultando de la siguiente manera:

```
[settings]
sipusers => mysql,general,sipfriends
sippeers => mysql,general,sipfriends
extensions => mysql,general,extensions
iaxusers => mysql,general,iax_buddies
iaxpeers => mysql,general, iax_buddies
voicemail =>mysql,general, voicemail
```

Como se puede ver, en la base de datos están almacenados los usuarios SIP (en la tabla *sipfriends*), las casillas de correo de voz de los mismos (tabla *voicemail*) y las extensiones que éstos pueden

marcar (tabla *extensions*) para realizar una llamada. Además, se incluyen dos túneles IAX (tabla *iax_buddies*) que se utilizan para la interconexión de los pares DUNDi (llamado *iaxuser*) y la conexión de los servidores de registro con el Gateway PSTN (llamado *pstn*). La configuración de estos túneles se muestra a continuación:

```
[pstn]
type=friend
context=internos
username=servidorreal
secret=password
auth=plaintext
host=192.168.1.252
trunk=yes
requirecalltoken=no
```

```
[iaxuser]
type=friend
dbsecret=dundi/secret
context=incomingdundi
```

Por otro lado, hizo falta agregar en el archivo *extensions.conf* la directiva *include* correspondiente para incluir las extensiones que se encuentran configuradas en la base de datos. Esto se realizó incluyendo la siguiente línea al final del contexto *internal*.

```
switch => Realtime
```

Cada vez que un servidor de registración debe ubicar a un usuario primero lo busca en forma local y si no lo encuentra envía una consulta al servidor de lookup, quien la propaga al resto de los servidores de registro y luego reenvía la respuesta obtenida a quien solicito la información. Esta operación de búsqueda remota se realiza a través de DUNDi, lo que solo requiere configurar al servidor de lookup como único par. De esta forma, la configuración del archivo *dundi.conf* quedó de la siguiente manera:

```
[general]

department=Lab Redes
organization=UNLPam
locality=General Pico, La Pampa
stateprov=LP
country=AR
email=asterisk@gmail.com
phone=+12565551212
bindaddr=192.168.1.162
port=4520
entityid= B8:AC:6F:68:A9:01
cachetime=5
ttl=2
autokill=yes

[mappings]
priv => dundiextens,0,IAX2,iaxuser:${SECRET}@192.168.1.162/${NUMBER},nopartial

[6C:F0:49:A4:4B:98] ; MAC Address NODO LOOKUP
model = symmetric
host = 192.168.1.169 ; IP address virtual del nodo Lookup
inkey = dundi
outkey = dundi
include = priv
permit = priv
qualify = yes
order = primary
```

En esta configuración se puede observar que en la sección Mappings se define lo que este servidor va a devolver como dirección de contacto cuando otro servidor le realice una consulta. Esto quiere decir que si una extensión SIP está registrada en este servidor, éste devolverá como respuesta a dicha consulta de búsqueda la ruta completa que deberá utilizar el servidor solicitante para contactarse con la extensión. Como también puede verse, todos estos intercambios de datos se realizan con el protocolo IAX2 a través de un túnel IAX, llamado *iaxuser,* que está configurado en la base de datos del servidor de lookup.

Cuando un usuario se registra en el servidor, se crea de forma dinámica la extensión para ubicar a este usuario, a partir de lo definido en la sección Mappings. Esta extensión se crea en un contexto llamado *incomingdundi*, el cual está especificado en el túnel IAX antes mencionado. A partir de esto, todas las llamadas entrantes para este usuario caerán dentro de este contexto y por lo tanto es necesario crearlo dentro del archivo *extensions.conf* como se muestra a continuación.

```
exten => _XXXX,1,NoOp(Llamada entrante al servidor de registro para ${EXTEN})
exten => _XXXX,n,Goto(internal,${EXTEN},2)
```

Además, cada vez que un usuario se registre en el servidor este es agregado en forma dinámica dentro del contexto *dundiextens*. Para lograr esto es necesario agregar en el archivo *sip.conf* lo siguiente.

```
[general]
regcontext=dundiextens
```

Bajo esta arquitectura utilizada fue necesario contar con una unidad de almacenamiento centralizada para el almacenamiento de mensajes de voz, ya que los usuarios deben poder acceder a sus mensajes independientemente del servidor en el que se encuentren registrados, manteniendo así la trasparencia a los usuarios de su ubicación en el sistema. Para lograrlo se implementó el servicio de directorios compartidos NFS el cual se ubicó en el servidor eurocase4 ya que este es un nodo central de la arquitectura y bajo situaciones normales de funcionamiento es el que presenta menor carga.

La configuración de este servicio se realiza principalmente a través de dos archivos de configuración, *exports* y *hosts.allow* cuyos correspondientes contenidos se muestran a continuación:

```
/var/spool/asterisk/voicemail 192.168.1.0/24(rw)
```

exports

```
portmap : 192.168.1.0/24
lockc : 192.168.1.0/24
mountd : 192.168.1.0/24
rquotad : 192.168.1.0/24
statd : 192.168.1.0/24
```

hosts.allow

Luego, en cada servidor de registro se agregó la línea correspondiente al archivo *fstab* para montar este directorio.

```
192.168.1.164:/var/spool/asterisk/voicemail /var/spool/asterisk/voicemail nfs
defaults 0 0
```

De esta forma se logró que todos los mensajes de correo de voz se almacenen en un servidor central y cualquier servidor de registro pueda acceder a los mismos.

7.3.4. Gateway PSTN

Debido a que la principal infraestructura implementada en este trabajo se enfocó puramente en tecnología VoIP se optó por utilizar un Gateway PSTN ya implementado. De esta forma, se intentó independizar la red principal al uso de cualquier Gateway disponible en el mercado. En el caso concreto de nuestra implementación se optó por el uso de otro servidor Asterisk ya implementado y con el hardware apropiado ya configurado para llevar a cabo esta funcionalidad. El mismo es el correspondiente al desarrollado en el proyecto final de Ingeniería en sistemas "IMPLEMENTACIÓN DE SERVICIOS DE VOIP UTILIZANDO ASTERISK" (Franco & Muller, 2009).

7.3.5. Extensiones de llamada

Para el caso práctico se crearon distintas extensiones para representar servicios y usuarios. Como se mencionó anteriormente, estas extensiones se encuentran almacenadas en la base de datos de los servidores de Lookup en la tabla *extensions*. Las principales extensiones se mencionan a continuación.

Servicio	Extensión	Prioridad	Aplicación	Parámetros
Servicio de Eco	100	1	Playback	Demo-echotest
	100	2	Echo	
Llamada entre usuarios DUNDi	_XXXX	2	NoOp	Llamada a ${EXTEN}
	_XXXX	3	Dial	SIP/${EXTEN},10
	_XXXX	4	Hangup	
Menú de correo de voz	864	1	VoiceMailMain	
	864	2	Hangup	
Correo de voz de la extensión	468	1	VoiceMailMain	${EXTEN}@eurocase
	468	2	Hangup	
Llamadas a PSTN (internos facultad)	_9XXXX	1	Dial	IAX2/pstn/${EXTEN:1},90,tr
	_9XXXX	2	Hangup	
Llamadas PSTN externas	_9	1	Dial	IAX2/pstn/${EXTEN},90,tr
	_9	2	Hangup	

7.4. Usuarios

Como se explicó en las secciones anteriores, los usuarios del sistema fueron configurados en la base de datos centralizada, en la tabla *sipfriends*. Se crearon diversos usuarios para probar el sistema, donde los principales datos de cada usuario contenidos en la tabla son el nombre, el tipo de usuario, el contexto al que pertenecen y los códecs que admiten.

Para probar el sistema se utilizaron distintos tipos de clientes. Se probó principalmente con softphones (teléfonos software ejecutados en las computadoras), aunque también se realizaron pruebas con teléfonos de voz IP. Además, se utilizó la herramienta SIPp para realizar pruebas exhaustivas del sistema, como así también monitoreo general de la infraestructura. Dada la complejidad y la importancia de esta herramienta a continuación se explica en profundidad el uso de la misma.

7.4.1. SIPp

Esta herramienta permite la creación de escenarios de prueba para poder testear el protocolo SIP. El propósito de su creación fue poder tener una herramienta que permita evaluar equipos reales como proxies SIP, servidores SIP, gateways SIP, PBX SIP, etc. como también emular una gran cantidad de agentes usuarios realizando llamadas simultaneas.

Como características posee la capacidad de generar tráfico SIP utilizando todos los métodos definidos específicamente para el protocolo (REGISTER, INVITE, BYE, SUBSCRIBE, ACK, CANCEL), manejando diferentes tráficos multimedia como audio y audio + video a través de RTP, comportándose indistintamente como un UAS (User Agent Server) o un UAC (User Agent Client). SIPp funciona sobre los protocolos de transporte UDP y TCP. La herramienta permite observar en tiempo de ejecución las estadísticas de las pruebas, por ejemplo call rate, round trip delay, duración de la llamada, retransmisiones de mensajes, etc., y ver en pantalla un gráfico con los mensajes especificados por el escenario, marco si fueron enviados o recibidos, la cantidad de veces que fueron enviados. Por otro lado, también permite la creación de logs de errores como de mensajes enviados/recibidos producidos durante la ejecución.

La configuración consiste esencialmente de dos tipos de archivos. Un archivo de configuración XML en donde se encuentra definido el escenario y un archivo de datos CSV donde se definen un

conjunto de valores que serán utilizados dentro del archivo XML en los campos especificados, como pueden ser usuarios, extensiones para llamadas, especificar datos de autentificación o cualquier otro valor que deba utilizarse en el escenario y deba especificarse de forma externa al escenario.

El archivo de configuración XML define el escenario que será ejecutado por SIPp, se compone de un conjunto de elementos que describen los mensajes SIP que deben ser enviados, las respuestas que deben o pueden recibirse y acciones que deben ejecutarse de acuerdo a los mensajes recibidos. Este archivo puede verse y estructurarse como si se tratara de un diagrama de flujo de un proceso computacional.

Es importante notar que la configuración de esta aplicación en principio no es complicada, aunque requiere un profundo conocimiento y entendimiento del protocolo SIP para la elaboración del archivo XML.

7.4.1.1. Estructura de los escenarios

Al utilizarse archivos XML para especificar escenarios, estos archivos deben seguir cumpliendo con las características que esto implica. Es decir, debe definirse el encabezado XML correspondiente con el tipo de codificación utilizada y toda la configuración debe realizarse dentro de un único elemento Raíz, que para SIPp el elemento Raíz obligatoriamente debe ser el elemento <scenario>. El elemento Raíz posee un único atributo, el atributo *name*, que define un nombre para el escenario. Estos archivos pueden ser validados con un esquema DTD creado para facilitar la creación de los escenarios.

7.4.1.2. Elementos del escenario

Dentro del elemento <scenario> se puede definir cualquiera de los siguientes elementos:

- **<send>:** Dentro de este elemento se describen los mensajes SIP que deben ser enviados a un servidor SIP o a un Agente de usuario cliente.

- **<recv>:** En este elemento se definen los mensajes de respuesta que se esperan del otro extremo de la conexión. A través del atributo *response* se indica que la respuesta debe ser un mensaje que contenga un código de respuesta (1xx, 2xx, 3xx, 4xx ó 5xx), en cambio si

se utiliza el atributo *request* se indica que el mensaje esperado es un requerimiento, por ejemplo un mensaje ACK.

- **<pause>:** Este elemento indica a través del atributo *milliseconds* el tiempo que SIPp debe esperar para continuar con la ejecución del escenario.

- **<nop>:** Este elemento no realiza nada a nivel SIP. Solo se utiliza para ejecutar en su interior alguna acción, por ejemplo un comando Shell.

- **<sendCmd>:** Este elemento se utiliza para enviar contenido si se está utilizando un escenario 3PPC (Third Party Call Control).

- **<recvCmd>:** Se utiliza para especificar los mensajes que deben ser recibidos en un escenario 3PPC.

- **<label>:** Este elemento se utiliza cuando se quiere pasar a secciones particulares del escenario. Se utiliza el atributo *id* para identificar estas secciones. Este atributo es un valor entero cuyo valor máximo que puede tomar es 19.

- **<Response Time Repartition>:** Especifica intervalos en milisegundos para distribuir los valores de respuesta.

- **<Call Length Repartition>:** Especifica intervalos en milisegundos para distribuir los valores para la medición de longitud de llamadas.

- **<Globals>:** Especifica nombre de variables con un alcance de tipo global.

- **<User>:** Especifica nombre de variables con un alcance de tipo usuario.

- **<Reference>:** Especifica nombre de variables para suprimir los mensajes de advertencia

por no ser utilizadas.

Cada uno de estos elementos posee atributos específicos para poder generar escenarios más completos y realizar pruebas más complejas.

Dependiendo de cuál sea el primer elemento que se encuentra en el escenario, este será interpretado por SIPp como un escenario UAC o UAS. Todo escenario que comienza con el elemento <send> es interpretado como un escenario UAC, en cambio si el primer elemento encontrado es el elemento <recv> el escenario es interpretado como un escenario UAS. De esta forma con SIPp puede probarse ambos lados del servicio.

7.4.1.3. Palabras claves o keywords

Dentro de la especificación de los mensajes que deben ser enviados o recibidos existen un conjunto de palabras claves para especificar ciertos valores que son generados por SIPp durante la ejecución de un escenario. Por ejemplo, números de secuencia de los mensajes, identificadores de llamadas, protocolo de transporte, etc. El listado completo de las palabras claves con sus significados se encuentra especificado en el Apéndice E.

7.4.1.4. Archivo de datos CSV

Los archivos CSV son documentos de formato abierto para representar en forma sencilla los valores de una tabla, donde el valor de cada columna es separada utilizando un caracter separador (que puede ser una coma o punto y coma) y cada salto de línea representa una fila de la tabla. Cuando el contenido mísmo posee comas o punto y coma es necesario utilizar un delimitador. Generalmente se utilizan las comillas para indicar que lo que se encuentra entre ellas es el valor de las columnas y así diferenciar los verdaderos separadores del contenido.

El uso de archivos de datos CSV en SIPp se realiza a través del parámetro –inf o utilizando el atributo *file* de la palabra clave *[field0-n]*. En la construcción de este archivo debe cumplirse con ciertas normas:

- La primera línea del archivo debe utilizarse únicamente para indicar la forma en que deben ser leídos los valores del archivo que se encuentran en las siguientes líneas:

- **SEQUENTIAL:** Cada línea del archivo será utilizada en forma secuencial en las consecutivas ejecuciones del escenario. De esta forma, la primera ejecución utilizará sólo la primera línea del archivo, la segunda línea para la segunda ejecución y así hasta completar todas las ejecuciones.

- **RANDOM:** Cada ejecución del escenario tomará en forma aleatoria una línea del archivo para ser utilizada en dicha ejecución.

- **USER:** Determina que el orden de las líneas es determinado por el usuario dentro del archivo XML del escenario utilizando las palabras claves antes mencionadas.

- Los valores de cada línea deben ser separados por punto y coma y se interpretan de la siguiente manera:

  [field0];[field1];[field2];[field3];…;[fieldn]

- Todas las líneas que comiencen con el carácter # serán interpretadas como comentarios.

7.4.2. Validación de escenarios

Al construirse los escenarios en el metalenguaje XML estos deben estar bien formados y deben cumplir con una estructura definida para que pueda ser interpretada por la herramienta. Para llevar a cabo esta realización, como la mayoría de los documentos XML, deben ser validados con un documento DTD que define la estructura que debe ser utilizada para mantener una consistencia y una estructura común entre todos los documentos. El DTD que define la estructura de los escenarios se puede encontrar en el Apéndice E.

7.4.2.1. Escenario de monitoreo

Para este trabajo se utilizó la herramienta SIPp para generar un escenario capaz de monitorear el servicio SIP en el nodo maestro a través del uso de dos operaciones fundamentales, la **registración** y la **llamada**. De esta forma podemos verificar que el servicio está plenamente en funcionamiento, donde los usuarios son capaces de registrarse en el servidor virtual y pueden comunicarse entre

ellos. Esto implica también el funcionamiento de la base de datos y la correcta comunicación con ésta.

A través del siguiente esquema se tratará de explicar el escenario planteado:

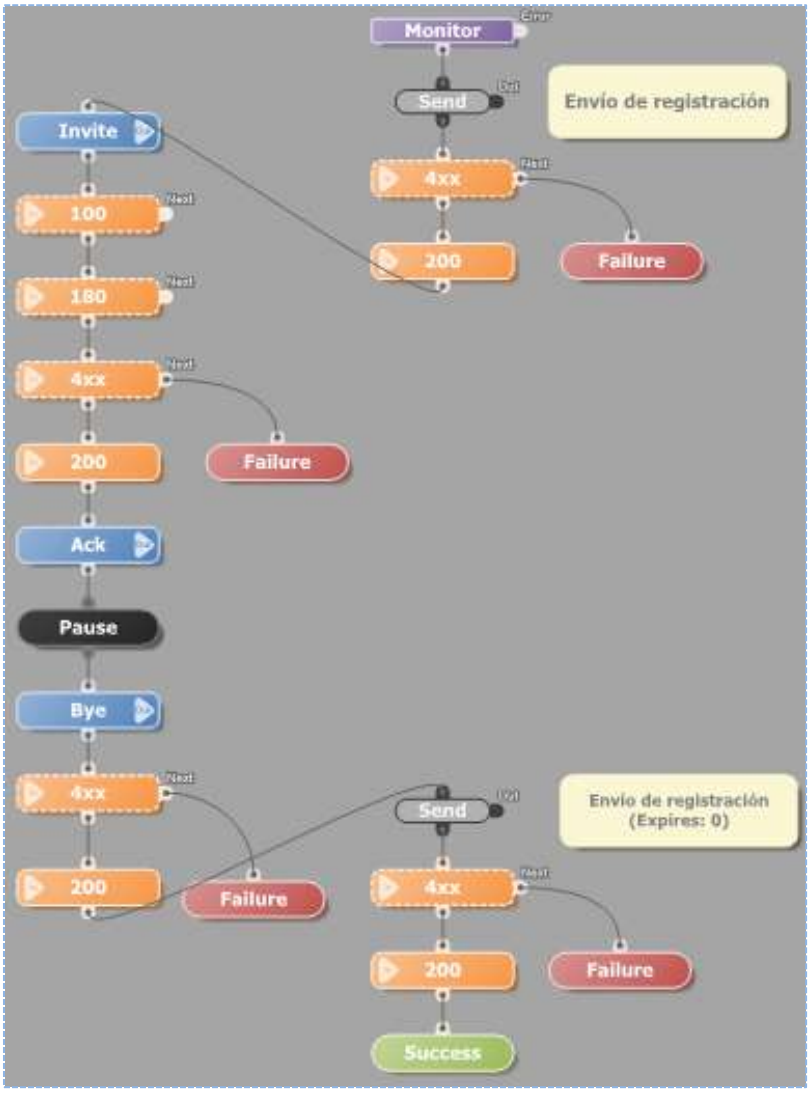


Figura 7.4.1: Esquema realizado con la herramienta SIPp_GUI.

El escenario puede ser separado en tres bloques:

- **Registración:** La primera acción del escenario es registrar un usuario creado en el servidor específicamente para realizar el monitoreo del servicio.

- Si se recibe algún código de error (4xx), el escenario es detenido. Estos mensajes de respuesta por parte del servidor Asterisk son declarados como mensajes opcionales, ya que serán recibidos únicamente si se produce algún error y su ausencia no implican la detención del escenario.

- **Llamada:** Cuando se recibe la confirmación de registración desde el servidor se envía el mensaje INVITE hacia el mismo llamando al servicio ECHO que posee Asterisk.
  - Cuando se inicia el proceso de llamada en SIP los códigos de mensaje que pueden ser recibidos por parte del servidor son: 100 TRYING, indica que el servidor está propagando la invitación hasta la ubicación del número destino; 180 RINGING, indica que el mensaje llegó hasta el extremo destino y está sonando hasta que sea atendida la comunicación; 4xx si se produjo algún error en el proceso de la invitación; y 200 – OK, cuando la llamada es atendida desde el otro extremo.
  - Los primeros 3 tipos de respuesta mencionados son declarados como opcionales, siendo los mensajes de error (4xx) los únicos capaces de cortar la ejecución del escenario.
  - La respuesta 200 – OK es declarada como una respuesta obligatoria, ya que debe aceptarse la llamada para poder continuar con el escenario.
  - Recibida la confirmación se envía un ACK al extremo de la comunicación, en este caso el servicio ECHO, indicando que la comunicación fue establecida en ambos sentidos.
  - Pasados 3 segundos de establecida la comunicación el escenario finaliza la llamada enviando el mensaje BYE.

- **Quitar registración:** Cuando se recibe la confirmación de finalización de la comunicación por parte del otro extremo se prosigue por eliminar la presencia del usuario de monitoreo en el servidor. Este paso es necesario para volver a ejecutar y evaluar el escenario completo, registración y llamada, en un próximo período de monitoreo del servicio Asterisk.

7.5. Puesta en marcha

Luego de haber explicado de qué forma están configurados los distintos componentes que forman parte de la infraestructura del clúster Asterisk se mostrará a continuación como se ven reflejadas estas configuraciones en el funcionamiento de la misma.

7.5.1. Servidores de lookup

En primer lugar, uno de los componentes principales de la infraestructura es el servidor de bases de datos MySQL. El mismo es accedido a través del módulo de Asterisk llamado RealTime y puede verificarse desde la consola de Asterisk si la comunicación con éste es exitosa. Esto se realiza mediante el comando "*realtime mysql status*" y la salida esperada debería ser como la Figura 7.5.1.

```
linux*CLI> realtime mysql status
general connected to asterisk@192.168.1.168, port 3306 with username asterisk for 2 hours, 58 minutes.
linux*CLI>
```

Figura 7.5.1: Estado de la conexión MySQL desde la consola Asterisk.

Una vez verificada la conexión, puede verse también cuales son las tablas que se obtienen desde la base de datos a través del comando "*realtime mysql cache*", en nuestro caso representada por la Figura 7.5.2.

```
linux*CLI> realtime mysql cache
             general iax_buddies
             general sipfriends
             general voicemail
linux*CLI> ~
```

Figura 7.5.2: Tablas accedidas por RealTime.

Otro componente importante de la infraestructura es el protocolo DUNDi, cuya implementación permite la localización de usuarios a lo largo de los servidores de registro. A través del comando "*dundi show peers*" desde la consola de Asterisk puede observarse si la configuración ésta correctamente realizada, cuáles son los pares a los que se conecta y el estado de dichas conexiones. En el caso de la infraestructura implementada para este proyecto, dado que existe un nodo central que concentra todas las consultas, los servidores de registro solo tienen configurado

ese nodo como único par. Por lo tanto la salida del comando debería ser similar a la mostrada en la Figura 7.5.3.

```
linux*CLI> dundi show peers
EID                 Host                Port   Model      AvgTime  Status
6c:f0:49:a4:4b:98   192.168.1.169   (S) 4520   Symmetric  Unavail  OK (1 ms)
1 dundi peers [1 online, 0 offline, 0 unmonitored]
linux*CLI>
```

Figura 7.5.3: Pares DUNDi de un servidor de registro.

Una vez funcionando DUNDi es posible realizar la búsqueda de cualquier usuario registrado en el sistema. Para probar esto se puede utilizar el comando *"dundi lookup <usuario>@<contexto>"*. En la Figura 7.5.4 se muestra la salida de este comando. Como se puede apreciar, la respuesta de este comando es la ruta que debe usarse para contactar al usuario buscado, incluyendo el protocolo, canal, password, servidor e interno (en dicho orden).

```
linux*CLI> dundi lookup 1066@priv
  1.     0 IAX2/iaxuser:8CXcJ+12UH8zBIt4dZE3Qg==@192.168.1.163/1066 (EXISTS)
     from b8:ac:6f:68:a8:9c, expires in 5 s
DUNDi lookup completed in 1 ms
linux*CLI>
```

Figura 7.5.4: Búsqueda del usuario 1066 en el contexto DUNDi llamado priv.

Este mapeo entre el contexto DUNDi y la ruta para localizar al usuario fue la configurada en el archivo *dundi.conf* y puede verificarse a través del comando *"dundi show mappings"* como se ve en la Figura 7.5.5.[3]

```
linux*CLI> dundi show mappings
DUNDi Cntxt  Weight  Local Cntxt  Options    Tech  Destination
priv         0       dundiextens  NONE       IAX2  iaxuser:${SECRET}@192.168
linux*CLI>
```

Figura 7.5.5: Correlación del contexto DUNDi con la ruta de contacto de usuarios.

7.5.2. Monitoreo

Otro aspecto importante de la infraestructura es el monitoreo constante de los servidores de registro por parte del balanceador de carga. El mismo se realiza, como ya se ha mencionado, utilizando SIPp. El mismo consiste en registrar un usuario (9999 o 1111 dependiendo del

[3] El campo Destination se imprime cortado por una limitación de la propia consola Asterisk. Debería ser de la forma iaxuser:${SECRET}@192.168.1.162/${NUMBER}.

balanceador que realiza la prueba), realizar una llamada al interno 100 (servicio de eco) y desregistrarse. A través del comando *"sip show peers"* puede verse cuales son los usuarios registrados en el servidor como lo muestra la Figura 7.5.6.

```
linux*CLI> sip show peers
Name/username             Host                                  Dyn Forcerport ACL Port     Status     Real
time
1066/1066                 (Unspecified)                          D   N      0        UNKNOWN    Cached RT
1111/1111                 192.168.1.161                          D   N      5070     OK (1 ms)  Cached RT
3002/3002                 192.168.1.67                           D   N      5060     OK (21 ms) Cached RT
9999/9999                 192.168.1.166                          D   N      5069     OK (1 ms)  Cached RT
sipp/sipp                 (Unspecified)                          D   N      0        Unmonitored Cached RT
5 sip peers [Monitored: 3 online, 1 offline Unmonitored: 0 online, 1 offline]
```

Figura 7.5.6: Usuarios registrados en el servidor de registro.

El servicio eco test antes mencionado, sirve para realizar una verificación de la latencia entre el teléfono del cliente y el servidor en el que se encuentra registrado. Además, siempre reproduce un sonido de bienvenida (denominado demo-ecotest). En la Figura 7.5.7 se muestra la salida retornada por la consola Asterisk durante la ejecución del monitoreo.

```
  -- Registered SIP '1111' at 192.168.1.161:5070
  -- Added extension '1111' priority 1 to dundiextens
== Using SIP RTP CoS mark 5
  -- Executing [100@internal:1] Playback("SIP/1111-00018d24", "demo-echotest")
  -- <SIP/1111-00018d24> Playing 'demo-echotest.gsm' (language 'en')
  -- Registered SIP '9999' at 192.168.1.166:5069
  -- Added extension '9999' priority 1 to dundiextens
== Using SIP RTP CoS mark 5
  -- Executing [100@internal:1] Playback("SIP/9999-00018d25", "demo-echotest")
  -- <SIP/9999-00018d25> Playing 'demo-echotest.gsm' (language 'en')
== Spawn extension (internal, 100, 1) exited non-zero on 'SIP/1111-00018d24'
== Spawn extension (internal, 100, 1) exited non-zero on 'SIP/9999-00018d25'
  -- Unregistered SIP '1111'
  -- Unregistered SIP '9999'
```

Figura 7.5.7: Proceso de monitoreo visualizado desde la consola Asterisk.

Durante el proceso del monitoreo descripto se intercambian numerosos paquetes del protocolo SIP. Estos incluyen el registro (REGISTER), la llamada (INVITE) y la des-registración (también como un paquete REGISTER) como puede observarse en la Figura 7.5.8.

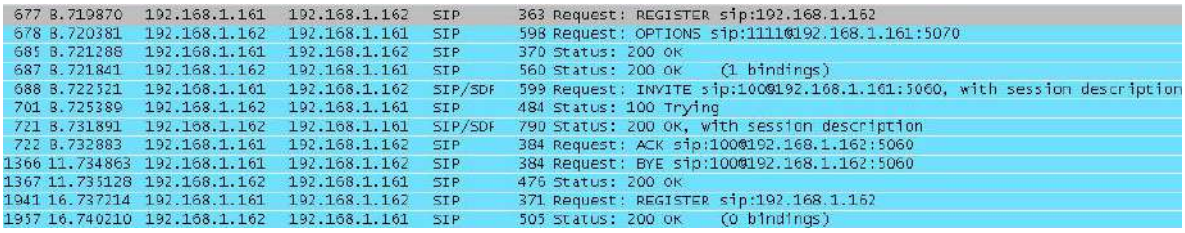


Figura 7.5.8: Intercambio de mensajes SIP durante el monitoreo.

Además, durante este proceso el servidor de registro debe consultar al servidor de base de datos acerca de toda la información relacionada a los usuarios y extensiones para llevar a cabo todas las operaciones involucradas. La Figura 7.5.9 muestra los paquetes intercambiados durante el monitoreo incluyendo dichas consultas. La Figura 7.5.10 muestra una de las consultas en detalle.

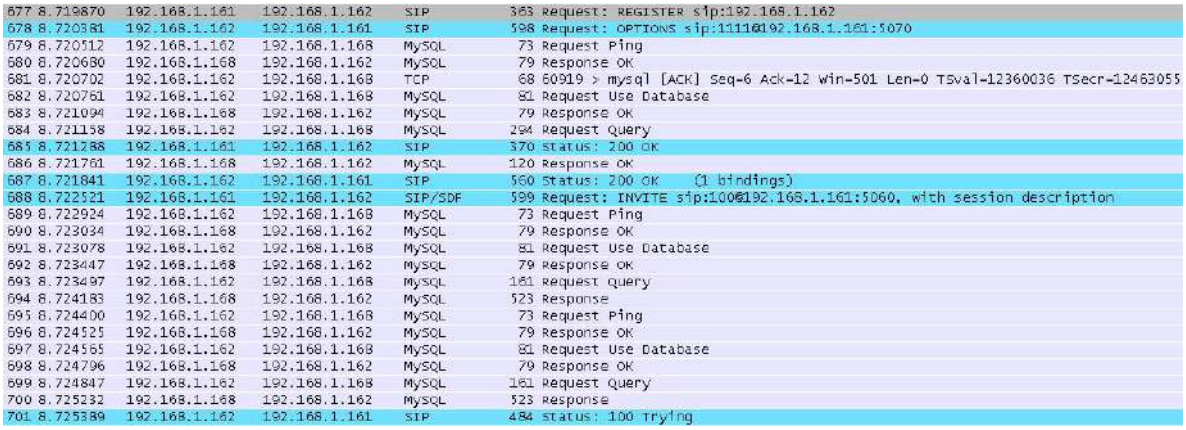


Figura 7.5.9: Intercambio de mensajes SIP y MySQL durante el monitoreo.

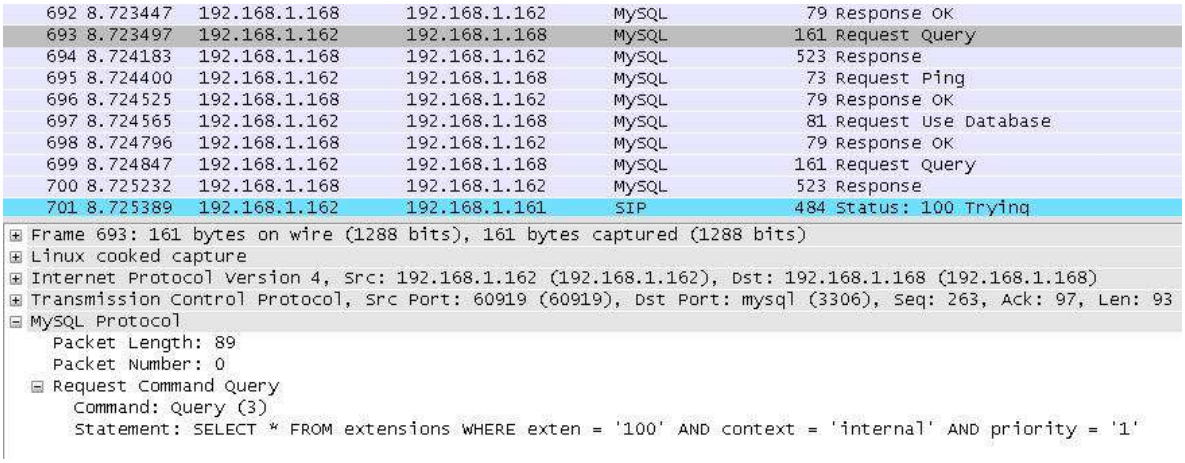


Figura 7.5.10: Detalle de un paquete MySQL.

7.5.3. Registración

Durante este proceso el cliente se contacta con una de las direcciones IP públicas en las cuales atienden los balanceadores de carga. Este último, recibe el paquete, realiza un encapsulamiento IP y reenvía el paquete a alguno de los servidores de registro. En la Figura 7.5.11 (frame número 5623) puede observarse que el cliente con dirección IP 192.168.1.67 envió el paquete SIP de registración a la dirección pública del balanceador (192.168.1.170), y éste encapsuló el encabezado IP y envió el paquete a través de su interfaz privada (192.168.1.166) hacia el servidor de registro *register.1* (192.168.1.162).

La Figura 7.5.12 (frame número 5624) muestra el mismo paquete mencionado anteriormente, pero esta vez desencapsulado, es decir tal como lo recibe el servidor de registro. A partir de la recepción de este paquete, el servidor puede realizar la respuesta del mismo directamente al cliente, como se puede ver en el frame número 5635 en cualquiera de las dos Figuras antes mencionadas.

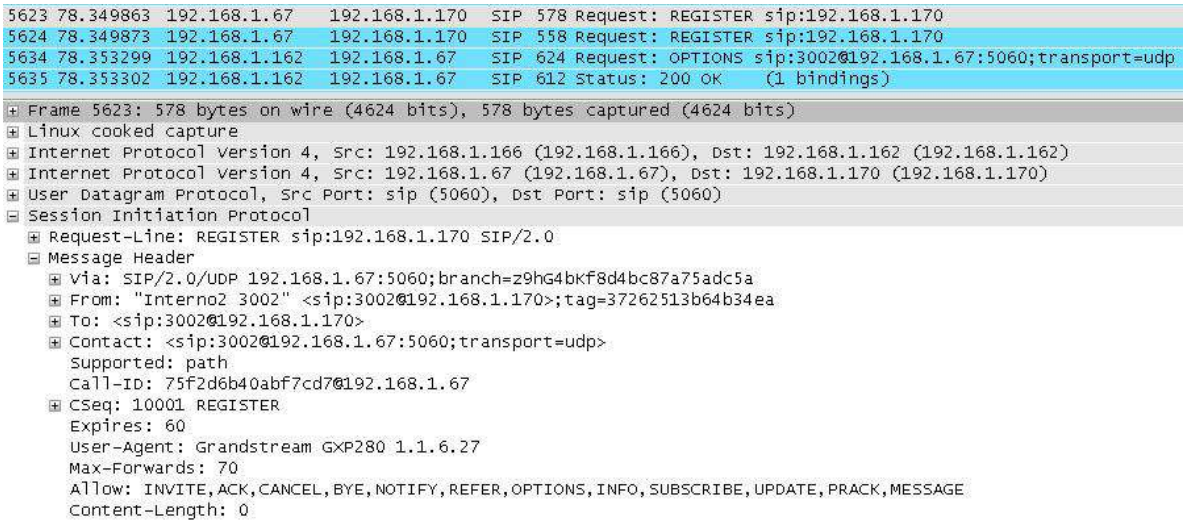


Figura 7.5.11: Encapsulamiento de paquete SIP por parte del balanceador.

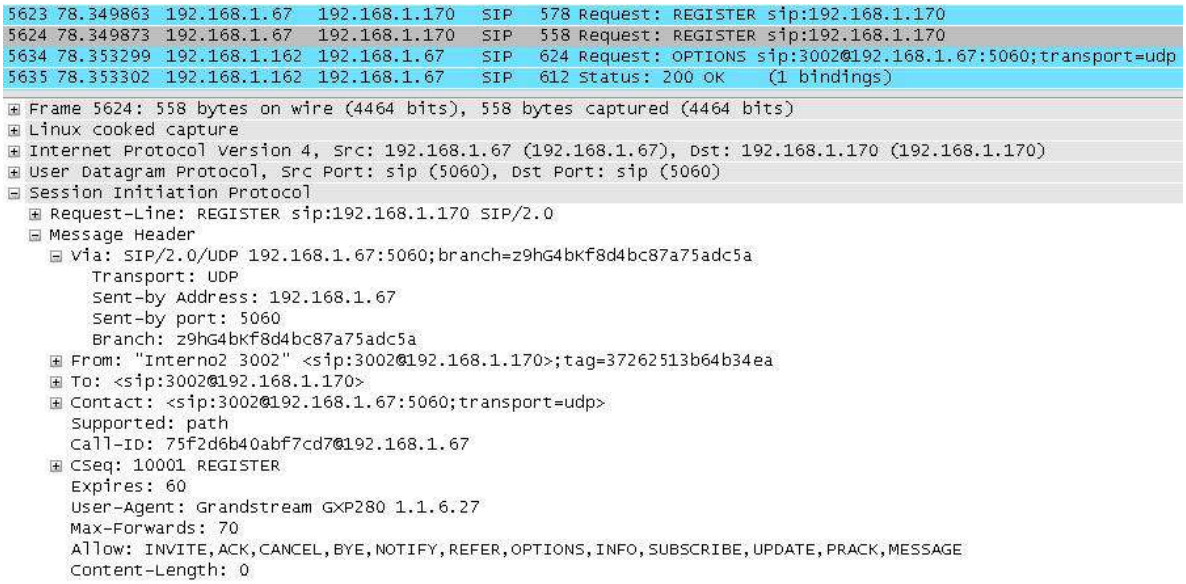


Figura 7.5.12: Des-encapsulamiento de paquete SIP por parte del servidor de registro.

7.5.4. Llamada

Para realizar llamadas, el cliente se debe contactar con el balanceador de carga, éste intermedia entre el cliente y el servidor de registro de forma similar a cómo se describió anteriormente para el proceso de registro. En el ejemplo a continuación, el cliente *"interno2 3002"* (192.168.1.67) realizó una llamada al interno 1066 ubicado en otro servidor de registro. Este proceso involucra encapsulamiento IP entre el balanceador y el servidor de registro, búsqueda del usuario 1066 utilizando DUNDi a través del servidor de lookup denominado lookup.1 con dirección IP 192.168.1.169, establecimiento de túnel IAX2 entre los servidores de registro y consultas a la base de datos MySQL.

En la Figura 7.5.13 (frame número 7853) se puede ver el primer paso del procedimiento del establecimiento de una llamada, la solicitud INVITE. En ésta se puede ver que el cliente con dirección IP 192.168.1.67 se contactó con el balanceador de carga a través de su interfaz pública (192.168.1.170). El mismo encapsuló el paquete y lo envió al servidor de registro register.1 (IP 192.168.1.162) a través de su interfaz privada (IP 192.168.1.166).

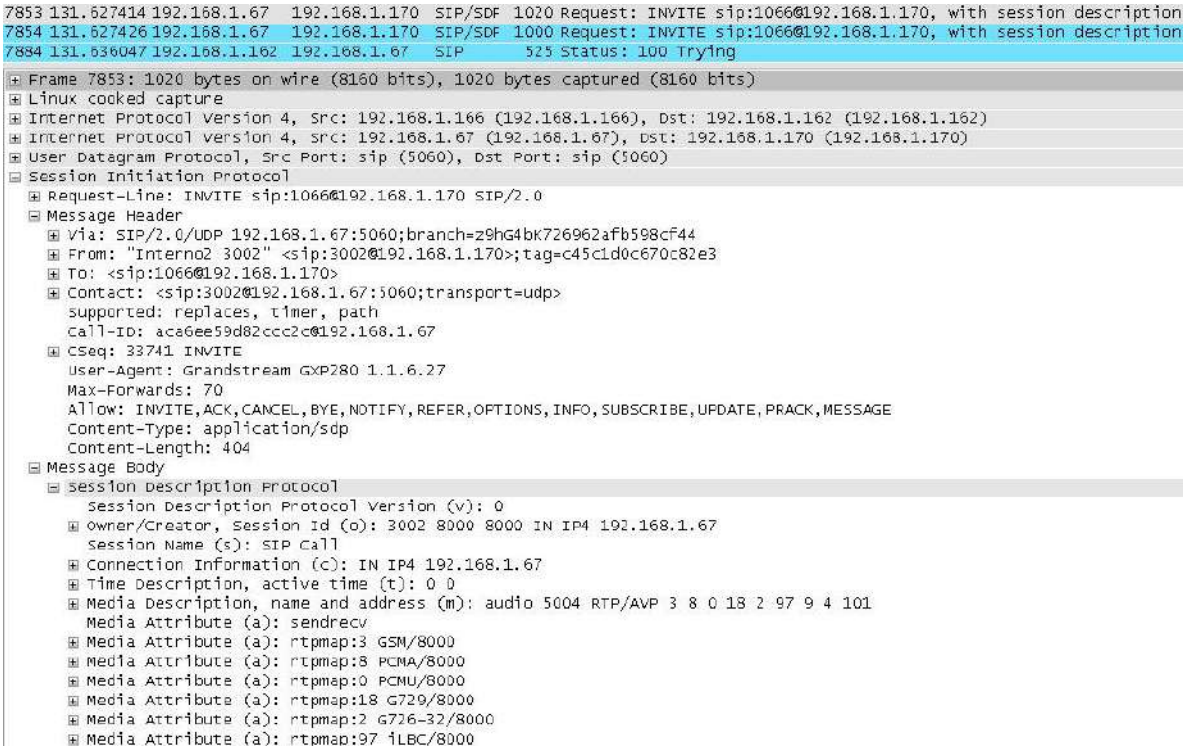


Figura 7.5.13: Encapsulamiento de inicio de llamada por parte del balanceador de carga.

La Figura 7.5.14 (frame número 7854) muestra el mismo paquete mencionado anteriormente pero desencapsulado por el servidor de registro.

Figura 7.5.14: Desencapsulamiento de inicio de llamada por parte del servidor de registro.

A partir de aquí, el establecimiento de la llamada se realiza en forma directa entre el servidor de registro y el cliente utilizando el protocolo SIP, como se muestra en la Figura 7.5.15.

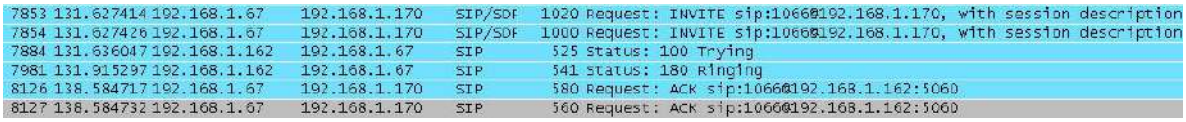
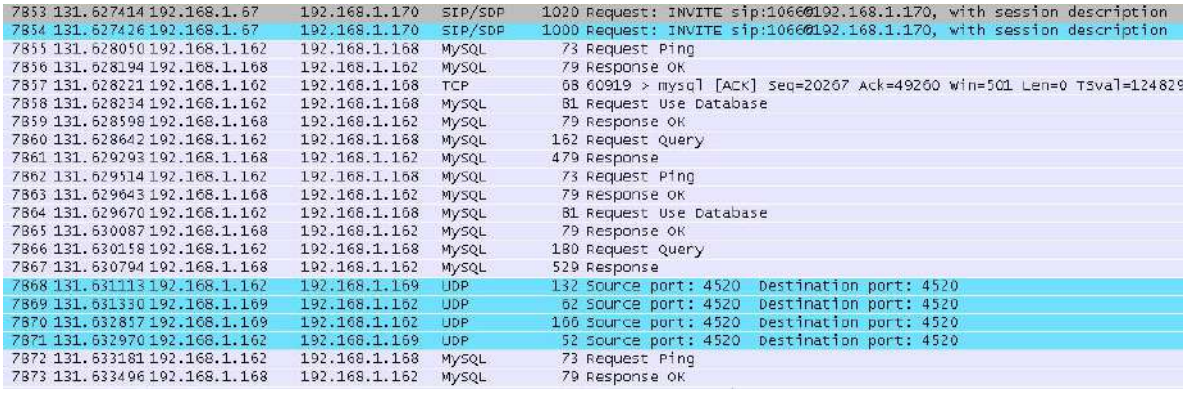


Figura 7.5.15: Establecimiento de una llamada.

La Figura 7.5.16 muestra más en detalle la figura anterior, incluyendo los paquetes intercambiados entre el servidor de registro y el servidor de bases de datos (IP 192.168.1.168) para la obtención de los datos de la extensión a la que se desea llamar (1066). Debido a que la extensión a la que se desea llamar se encuentra en otro servidor de registro, se realiza las correspondientes consultas a través de DUNDi utilizando como intermediario el servidor de lookup llamado lookup.1 (IP 192.168.1.169). En la figura, éstos son los paquetes comprendidos entre los frames 7868 al 7871.



7853	131.627414	192.168.1.67	192.168.1.170	SIP/SDP	1020	Request: INVITE sip:10660192.168.1.170, with session description
7854	131.627426	192.168.1.67	192.168.1.170	SIP/SDP	1000	Request: INVITE sip:10660192.168.1.170, with session description
7855	131.628050	192.168.1.162	192.168.1.168	MySQL	73	Request Ping
7856	131.628194	192.168.1.168	192.168.1.162	MySQL	79	Response OK
7857	131.628221	192.168.1.162	192.168.1.168	TCP	68	60919 > mysql [ACK] Seq=20267 Ack=49260 Win=501 Len=0 TSval=124829
7858	131.628234	192.168.1.162	192.168.1.168	MySQL	81	Request Use Database
7859	131.628598	192.168.1.168	192.168.1.162	MySQL	79	Response OK
7860	131.628642	192.168.1.162	192.168.1.168	MySQL	162	Request Query
7861	131.629293	192.168.1.168	192.168.1.162	MySQL	479	Response
7862	131.629514	192.168.1.162	192.168.1.168	MySQL	73	Request Ping
7863	131.629643	192.168.1.168	192.168.1.162	MySQL	79	Response OK
7864	131.629670	192.168.1.162	192.168.1.168	MySQL	81	Request Use Database
7865	131.630087	192.168.1.168	192.168.1.162	MySQL	79	Response OK
7866	131.630158	192.168.1.162	192.168.1.168	MySQL	180	Request Query
7867	131.630794	192.168.1.168	192.168.1.162	MySQL	529	Response
7868	131.631113	192.168.1.162	192.168.1.169	UDP	132	Source port: 4520 Destination port: 4520
7869	131.631330	192.168.1.169	192.168.1.162	UDP	62	Source port: 4520 Destination port: 4520
7870	131.632857	192.168.1.169	192.168.1.162	UDP	166	Source port: 4520 Destination port: 4520
7871	131.632970	192.168.1.162	192.168.1.169	UDP	52	Source port: 4520 Destination port: 4520
7872	131.633181	192.168.1.162	192.168.1.168	MySQL	73	Request Ping
7873	131.633496	192.168.1.168	192.168.1.162	MySQL	79	Response OK

Figura 7.5.16: Establecimeinto de una llamada.

Para el intercambio de datos entre los dos servidores de registro involucrados (register.1 IP 192.168.1.162 y register.2 IP 192.168.1.163) se establece un túnel utilizando el protocolo IAX2. En la Figura 7.5.17 se observa la creación de dicho túnel (frame 7956) y el proceso de autenticación involucrado en el mismo (frames 7970 al 7973). Una vez establecido el túnel, se realiza una llamada entre los dos servidores a través de éste túnel. Por otro lado, el otro servidor de registro establece una llamada con el cliente destino utilizando SIP. El procedimiento es simétrico al mostrado en estas capturas y por lo tanto no se ha incluido en el trabajo.

Una vez establecida la llamada entre el servidor de registro y el destinatario, se establece una conexión RTP para el intercambio de datos multimedia entre los servidores de registro y sus correspondientes clientes. Estos datos multimedia son intercambiados entre los servidores de registro a través del túnel IAX antes establecido, pero utilizando paquetes mini-frame (por ejemplo frame 8124).

7854	131.627426	192.168.1.67	192.168.1.170	SIP/SDP	Request: INVITE sip:1066@192.168.1.170, with session description
7868	131.631113	192.168.1.162	192.168.1.169	UDP	Source port: 4520 Destination port: 4520
7869	131.631330	192.168.1.169	192.168.1.162	UDP	Source port: 4520 Destination port: 4520
7870	131.632857	192.168.1.169	192.168.1.162	UDP	Source port: 4520 Destination port: 4520
7871	131.632970	192.168.1.162	192.168.1.169	UDP	Source port: 4520 Destination port: 4520
7884	131.636047	192.168.1.162	192.168.1.67	SIP	Status: 100 Trying
7956	131.667200	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 4ms NEW
7957	131.667522	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 1, timestamp 4ms CALLTOKEN
7958	131.667661	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 5ms NEW
7970	131.671564	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 6, timestamp 6ms AUTHREQ
7971	131.671782	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 10ms AUTHREP
7972	131.672096	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 6, timestamp 7ms ACCEPT
7973	131.672226	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 7ms ACK
7975	131.699740	192.168.1.169	192.168.1.162	UDP	Source port: 4520 Destination port: 4520
7976	131.701787	192.168.1.162	192.168.1.169	UDP	Source port: 4520 Destination port: 4520
7977	131.701973	192.168.1.162	192.168.1.169	UDP	Source port: 4520 Destination port: 4520
7978	131.702366	192.168.1.169	192.168.1.162	UDP	Source port: 4520 Destination port: 4520
7979	131.915031	192.168.1.163	192.168.1.162	IAX2	Control, source call# 6, timestamp 250ms RINGING
7980	131.915108	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 250ms ACK
7981	131.915297	192.168.1.162	192.168.1.67	SIP	Status: 180 Ringing
8091	138.549806	192.168.1.163	192.168.1.162	IAX2	Control, source call# 6, timestamp 6885ms ANSWER
8092	138.549821	192.168.1.163	192.168.1.162	IAX2	Control, source call# 6, timestamp 6888ms stop sounds
8093	138.549824	192.168.1.163	192.168.1.162	IAX2	Control, source call# 6, timestamp 6891ms unknown (0x14)
8094	138.549937	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 6885ms ACK
8095	138.550063	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 6888ms ACK
8097	138.550233	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 6891ms ACK
8098	138.550378	192.168.1.162	192.168.1.67	SIP/SDP	Status: 200 OK, with session description
8099	138.550504	192.168.1.162	192.168.1.163	IAX2	Control, source call# 1109, timestamp 6888ms unknown (0x14)
8100	138.550570	192.168.1.162	192.168.1.163	IAX2	Control, source call# 1109, timestamp 6891ms unknown (0x14)
8101	138.550595	192.168.1.162	192.168.1.163	IAX2	Control, source call# 1109, timestamp 6894ms unknown (0x14)
8102	138.550601	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 6, timestamp 6888ms ACK
8104	138.550699	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 6, timestamp 6891ms ACK
8106	138.550707	192.168.1.163	192.168.1.162	IAX2	IAX, source call# 6, timestamp 6894ms ACK
8121	138.556084	192.168.1.163	192.168.1.162	IAX2	Voice, source call# 6, timestamp 6897ms, GSM compression
8122	138.556165	192.168.1.162	192.168.1.163	IAX2	IAX, source call# 1109, timestamp 6897ms ACK
8123	138.556257	192.168.1.162	192.168.1.67	RTP	PT=GSM 06.10, SSRC=0x793C1BF2, Seq=48412, Time=160, Mark
8124	138.565989	192.168.1.163	192.168.1.162	IAX2	Mini packet, source call# 6, timestamp 6917ms, GSM compression
8125	138.566153	192.168.1.162	192.168.1.67	RTP	PT=GSM 06.10, SSRC=0x793C1BF2, Seq=48413, Time=320
8127	138.584732	192.168.1.67	192.168.1.170	SIP	Request: ACK sip:1066@192.168.1.162:5060
8128	138.586106	192.168.1.163	192.168.1.162	IAX2	Mini packet, source call# 6, timestamp 6937ms, GSM compression
8129	138.586245	192.168.1.162	192.168.1.67	RTP	PT=GSM 06.10, SSRC=0x793C1BF2, Seq=48414, Time=480
8130	138.604112	192.168.1.67	192.168.1.162	RTP	PT=GSM 06.10, SSRC=0x17BA182B, Seq=39828, Time=2508887454

Figura 7.5.17: Comunicación entre servidores de registro para el establecimiento de una llamada.

Desde la consola de Asterisk todo este proceso para el establecimiento de la llamada se visualiza de forma similar a como se muestra en la Figura 7.5.18 . En la misma se observa la calidad de servicio (CoS mark 5) con la cual se establece el envío de los paquetes RTP. Además, se puede apreciar que el protocolo utilizado para la comunicación entre los servidores de registro es IAX2 como se mencionó anteriormente y que el códec de audio elegido para la llamada es GSM.

```
  == Using SIP RTP CoS mark 5
    -- Called IAX2/iaxuser:8CXcJ+12UH8zBIt4dZE3Qg==@192.168.1.163/1066
    -- Call accepted by 192.168.1.163 (format gsm)
    -- Format for call is gsm
    -- IAX2/192.168.1.163:4569-1698 is ringing
    -- IAX2/192.168.1.163:4569-1698 answered SIP/3002-00018d78
    -- Started music on hold, class 'default', on IAX2/192.168.1.163:4569-1698
    -- Stopped music on hold on IAX2/192.168.1.163:4569-1698
    -- Hungup 'IAX2/192.168.1.163:4569-1698'
  == Spawn extension (internal, 1066, 1) exited non-zero on 'SIP/3002-00018d78'
linux*CLI>
```

Figura 7.5.18: Consola Asterisk al momento de una llamada.

7.6. Pruebas y análisis de resultados

Para evaluar la funcionalidad del sistema y poner a prueba la capacidad del mismo se realizaron distintas pruebas exhaustivas utilizando SIPp. Las mismas consistieron en realizar una gran cantidad de llamadas simultáneas, utilizando de dos a tres clientes donde cada uno estableció una gran cantidad de conexiones para poder ser distribuidas por el balanceador. Para analizar el comportamiento del clúster se llevaron a cabo diversas mediciones tanto en los servidores de registro como en el balanceador. A estas pruebas las llamaremos *"Prueba 9000"* y *"Prueba 12000"* en referencia a la cantidad de llamadas concurrentes que efectuó cada cliente respectivamente.

Por otra parte, se realizaron pruebas para evaluar la disponibilidad de la infraestructura y determinar la magnitud que puede significar la pérdida de alguno de sus componentes.

7.6.1. Pruebas con SIPp

Las pruebas consistieron en realizar hasta nueve mil y doce mil llamadas simultáneas desde dos y tres clientes SIPp hacia los dos balanceadores. Se configuró cada cliente para que abra un máximo de mil sockets UDP para realizar llamadas (por una limitación de los equipos usados como clientes) los cuales son reusados en caso de que las llamadas concurrentes superen las mil. Esto además, permitió que los balanceadores de carga pudieran distribuir las conexiones entre los distintos servidores de registro, debido a que los balanceadores balancean conexiones a nivel transporte y no llamadas a nivel aplicación.

Cada cliente realizó llamadas a la extensión cien, perteneciente al servicio de eco, a una velocidad de quince por segundo en *"Prueba 9000"* y treinta por segundo en *"Prueba 12000"*, con una duración máxima de doscientos segundos cada una. Las pruebas fueron realizadas durante diez minutos. Luego de este lapso se detuvo a los clientes y se esperó cinco minutos aproximadamente hasta que finalizaron todas las llamadas en progreso y expiraron las conexiones en los balanceadores.

Para lograr esto se ejecutó por consola en cada uno de los equipos clientes el siguiente comando:

```
sipp -sn uac <DIRECCION_IP> -d 200000 -l <LIMITE> -s 100 -t un \
      -max_socket 1000 -r <VELOCIDAD>
```

Dónde:

- **sn:** Establece el escenario utilizado (ya embebido en la herramienta SIPp). El valor *uac* representa al estándar SipStone UAC.
- **<DIRECCION_IP>:** Establece la dirección IP a la cual se realizan las llamadas. Fue reemplazado por alguna de las dos direcciones IP públicas de los balanceadores de carga.
- **d:** Establece la duración de la llamada en milisegundos.
- **l:** Establece el número límite de llamadas simultáneas. En *"Prueba 9000"* el valor utilizado fue 3000 con tres clientes y en *"Prueba 12000"* el valor fue 6000 con dos clientes.
- **s:** Establece la extensión destino de la llamada.
- **t:** Establece el modo de transporte, el valor *un* especifica un socket UDP por llamada.
- **max_socket:** Establece la canitadad máxima de sockets que se pueden utilizar. Si se alcanza este límite las llamadas son distribuidas entre los sockets ya abiertos.
- **r:** Establece la cantidad de llamadas efectuadas por segundo. En *"Prueba 9000"* el valor utilizado fue 15 con tres clientes y en *"Prueba 12000"* el valor fue 30 con dos clientes.

Adicionalmente, esta herramienta permite ir visualizando en tiempo real la operación. La Figura 7.6.1 muestra una captura realizada de la salida perteneciente a la primera prueba. En la misma se puede observar que la herramienta proporciona información detallada acerca de los mensajes enviados, retransmisiones y errores. También muestra la cantidad de llamadas concurrentes, cuanto se tardó en alcanzar el límite establecido y la cantidad de sockets en uso. La Figura 7.6.2 muestra la misma salida correspondiente a *"Prueba 12000"*.

Durante todas estas pruebas se comprobó que los servidores de registro y la infraestructura en general respondían perfectamente a las funciones que debían desempeñar. Se llevaron a cabo pruebas adicionales con exigencias mayores y menores a las mencionadas en este proyecto con el fin de determinar cuáles eran los límites soportados por la infraestructura y se comprobó que los parámetros utilizados en *"Prueba 12000"* son los que determinan la carga máxima que ésta soporta.

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
0.0(200000 ms)/1.000s   5060     891.54 s         9090  192.168.1.167:5060(UDP)

  0 new calls during 1.002 s period      1 ms scheduler resolution
  0 calls (limit 3000)                   Peak was 3000 calls, after 200 s
  0 Running, 2 Paused, 3 Woken up
  0 dead call msg (discarded)            0 out-of-call msg (discarded)
  3 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->        9090      0         0
         100 <----------        9090      0         0         0
         180 <----------        0         0         0         0
         183 <----------        0         0         0         0
         200 <----------  E-RTD1 9090      0         0         0
         ACK ---------->        9090      0
       Pause [     3:20]        9090                          0
         BYE ---------->        9090      0         0
         200 <----------        9090      0         0         0

------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----
```

Figura 7.6.1: Salida devuelta en Prueba 9000 por uno de los clientes.

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
0.0(200000 ms)/1.000s   5060     720.32 s        18017  192.168.1.167:5060(UDP)

  0 new calls during 1.001 s period      1 ms scheduler resolution
  2412 calls (limit 6000)                Peak was 6000 calls, after 200 s
  0 Running, 3404 Paused, 62 Woken up
  0 dead call msg (discarded)            0 out-of-call msg (discarded)
  1000 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->        18017     0         0
         100 <----------        18017     0         0         0
         180 <----------        0         0         0         0
         183 <----------        0         0         0         0
         200 <----------  E-RTD1 18017     0         0         0
         ACK ---------->        18017     0
       Pause [     3:20]        18017                         0
         BYE ---------->        15605     0         0
         200 <----------        15605     0         0         0

------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----
```

Figura 7.6.2: Salida devuelta en Prueba 12000 por uno de los clientes.

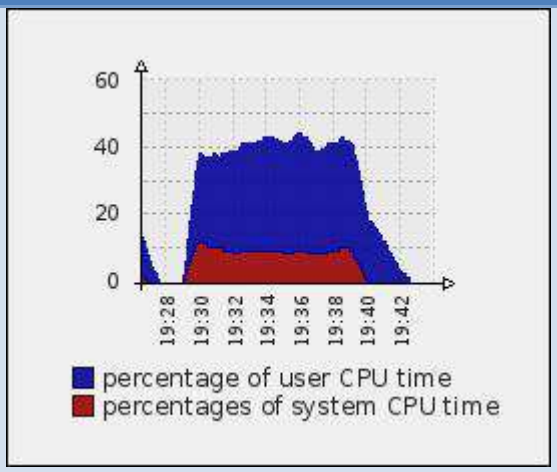
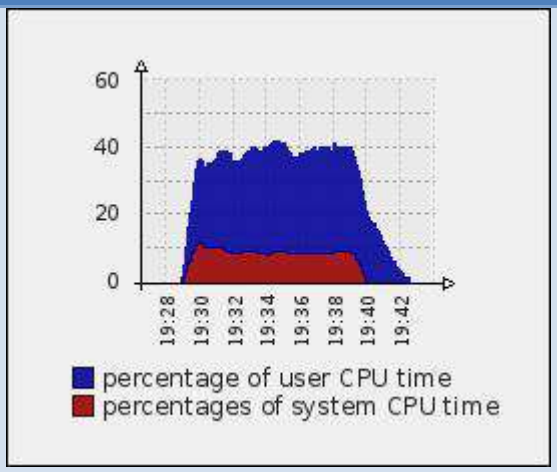
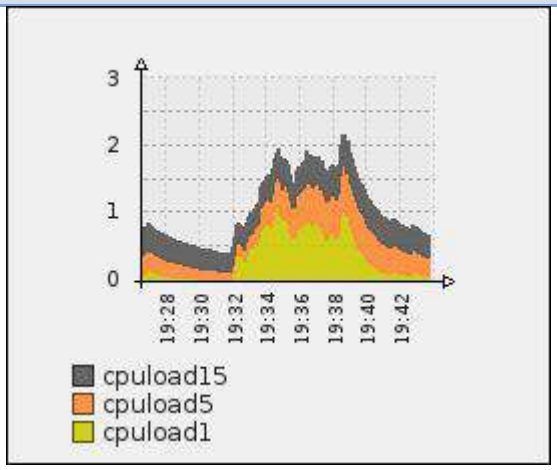
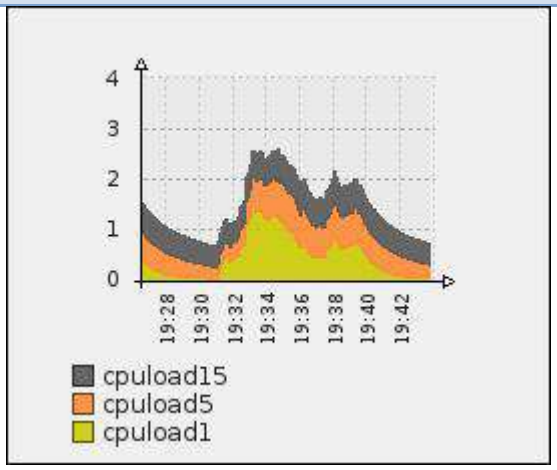
A continuación se detallan algunos gráficos obtenidos del monitoreo de los servidores durante las pruebas realizadas (*"Prueba 9000"* y *"Prueba 12000"*). Los mismos sirven como referencia para analizar el desempeño de los servidores de registro ante los distintos escenarios planteados.

Referencias a tener en cuenta de los gráficos a continuación:

- **cpuload1, cpuload5, cpuload15:** Se refiere a la carga de CPU registrada al minuto, a los 5 minutos y a los 15 minutos transcurridos respectivamente de haber iniciado la medición.

- **IfOutOctets2:** Se refiere a la cantidad de paquetes enviados a través de la interfaz de red.

- **IfInOctets2:** Se refiere a la cantidad de paquetes recibidos desde la interfaz de red.

Register.1	Register.2
	
Uso de CPU.	**Uso de CPU.**
	
Carga de CPU.	**Carga de CPU.**

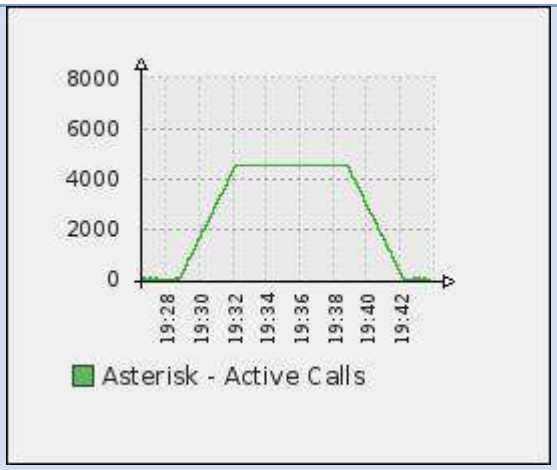
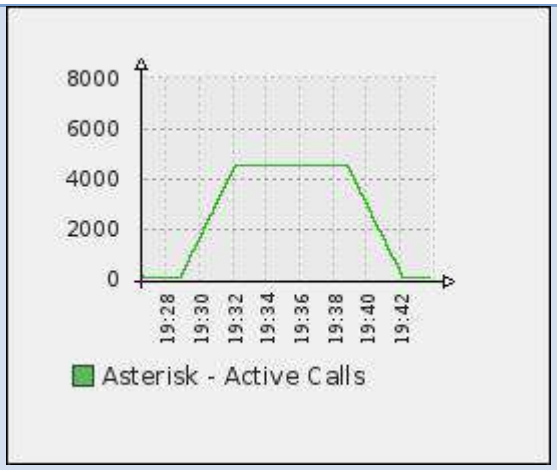
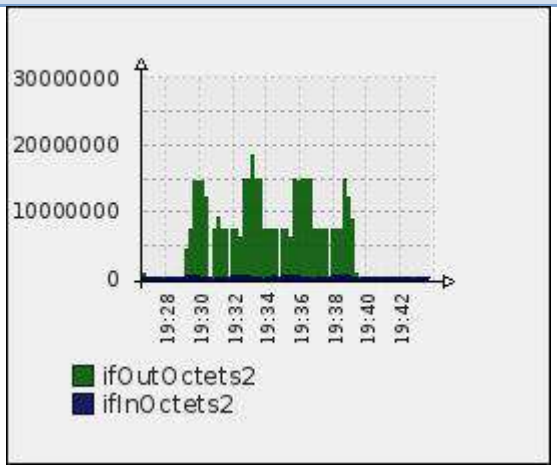
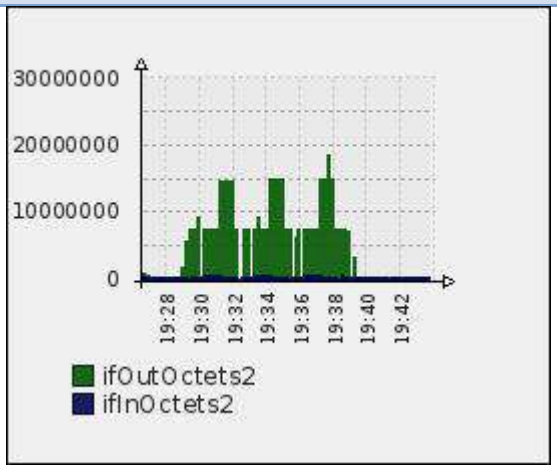
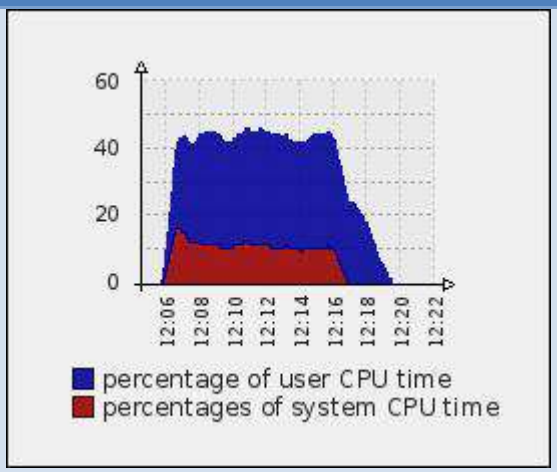
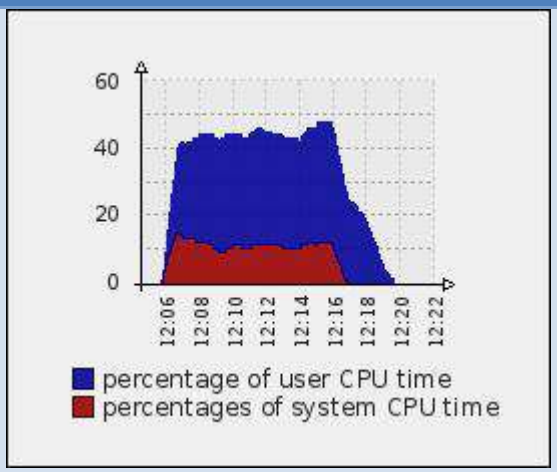
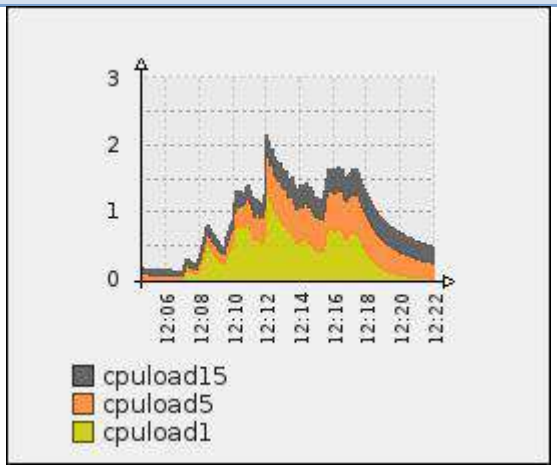
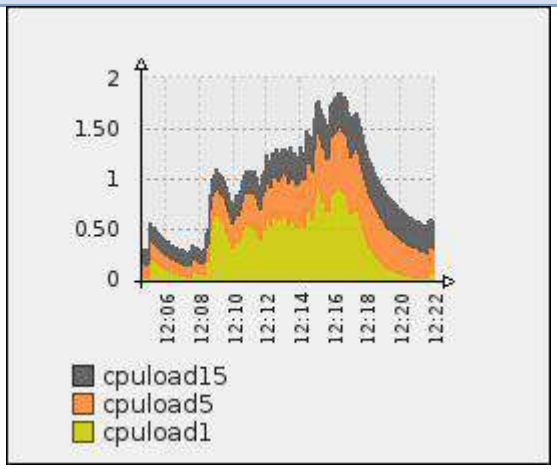
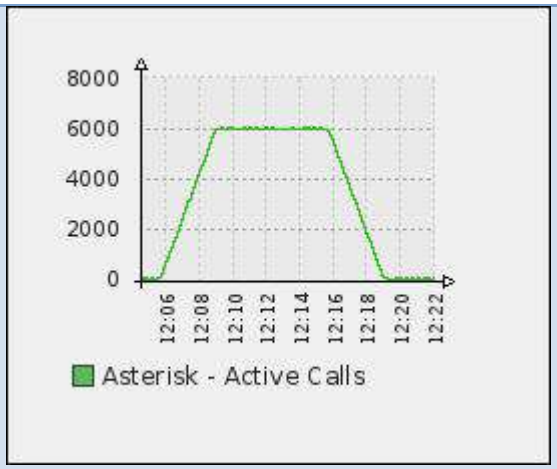
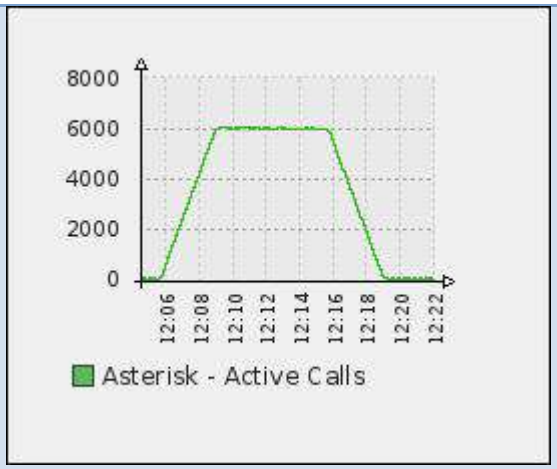
	
Llamadas activas en Asterisk.	**Llamadas activas en Asterisk.**
	
Utilización de red.	**Utilización de red.**

Tabla 7.6.1: Gráficos correspondientes a *"Prueba 9000"*.

Register.1	Register.2
	
Uso de CPU.	**Uso de CPU.**
	
Carga de CPU.	**Carga de CPU.**
	
Llamadas activas en Asterisk.	**Llamadas activas en Asterisk.**

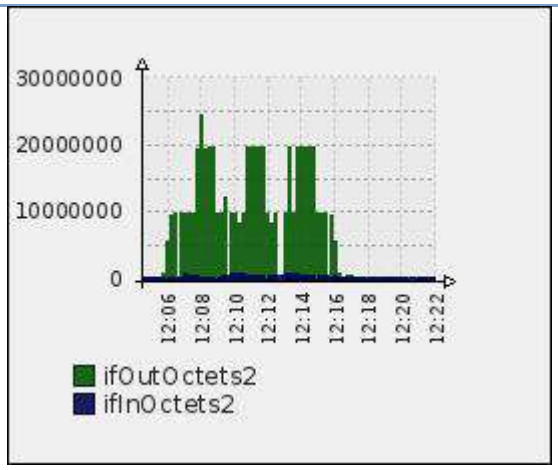
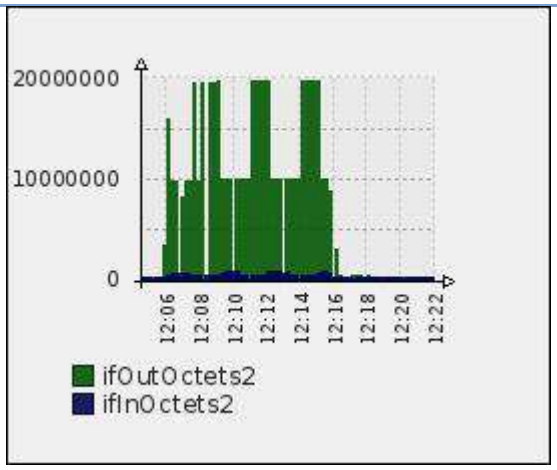
	
Utilización de red.	**Utilización de red.**

Tabla 7.6.2: Gráficos correspondientes a *"Prueba 12000"*

7.6.2. Pruebas de disponibilidad

Se puso a prueba la disponibilidad del servicio deteniendo alguno de los balanceadores de carga durante el transcurso de una llamada activa. Se pudo comprobar que esta baja no afecta en lo más mínimo la llamada mencionada, permitiendo que la misma pueda llevarse a cabo y finalizar sin problemas. Esto se debe a que una vez establecida la llamada, el cliente se comunica en forma directa con el servidor de registro evitando pasar por el balanceador.

Por otro lado, se comprobó mediante pruebas que si se cae o da de baja uno de los servidores de registro, la misma es detectada por el balanceador de carga a través de su sistema de monitoreo y todos los clientes registrados en este servidor son redirigidos dentro de los próximos sesenta segundos a cualquier otro servidor de registro activo. Esto es posible gracias a que los clientes están configurados para renovar la registración cada sesenta segundos.

Es importante mencionar que si alguno de los servidores de registro deja de responder durante el transcurso de una llamada activa, la misma termina de forma inmediata. Resulta imposible desde el punto de vista de la infraestructura evitar este caso, ya que el servidor de registro interviene de forma activa en todas las llamadas. Sin embargo, si el cliente vuelve a realizar la llamada inmediatamente la misma puede ser concretada.

Otro de los casos que se probó fue la simulación de la pérdida de alguno de los servidores de lookup. En este caso, dado que están replicados la baja resulta prácticamente transparente para la

infraestructura. En cuestión de segundos Heartbeat detecta la ausencia del mismo y migra la dirección IP virtual al servidor restante, migrando el servicio si también corresponde. Si el servidor afectado es el que ejecuta el servicio de base de datos, la integridad de los datos de los usuarios y las extensiones están a salvo ya que el mismo se encuentra replicado. Como la replicación configurada funciona como master-master, el proceso de recuperación del servidor caído tampoco se ve afectado ya que eventualmente actualizaría los datos con información más reciente. Si el servidor de lookup afectado es el que funciona como concentrador DUNDi, el mismo es iniciado en el servidor restante e inmediatamente comienza a proveer el servicio sin la necesidad de un procesamiento extra.

Por último, si el servidor afectado llegase a ser el servidor Gateway, el servicio de interconexión con la red PSTN se vería afectado por no estar replicado aunque podría haber más de uno funcionando. En este caso, el resto de los servicios de telefonía seguirían funcionando sin problemas.

7.7. Conclusiones de las pruebas

A partir de las pruebas realizadas se pudo observar que la infraestructura se comportó de la forma esperada, pudiendo satisfacer el máximo de doce mil llamadas activas simultáneas entre dos servidores de registro.

Por un lado se comprobó que los balanceadores de carga distribuyeron correctamente todas las conexiones de forma esperada a la política de balanceo aplicada, y sincronizaron entre ellos todas las conexiones activas. A partir de esto se puede afirmar que los balanceadores no resultan ser un factor limitante para la infraestructura en cuanto a cantidad de conexiones concurrentes que pueden atender los servidores de registro.

A partir de los gráficos de monitoreo se detectó que la carga máxima de cualquiera de los servidores de registro se mantiene prácticamente por debajo de 1.0, lo cual quiere decir que todos los procesos son atendidos a tiempo y por lo tanto no se detectan problemas de latencia ni cortes en las comunicaciones. Por ello se puede afirmar que el factor limitante resulta ser, como era de esperarse, las características técnicas del servidor de registro. Sin embargo, la infraestructura está

diseñada para poder sobrellevar este problema rápidamente agregando más servidores de registro según sea necesario.

Teniendo en cuenta que la cantidad de llamadas activas simultaneas es de doce mil, y considerando que en una población de un millón de habitantes bajo condiciones normales solo el 10% de los usuarios realizan llamadas concurrentemente, se puede estimar que una infraestructura como la presentada en este proyecto podría llegar a soportar entre 100.000 y 120.000 clientes.

Se debe tener en cuenta que tener solo dos servidores de registro resulta relativamente riesgoso en un escenario como el planteado anteriormente, ya que ante la caída de uno de ellos todos los usuarios del mismo serían migrados al servidor restante y al mismo le resultaría imposible atender las doce mil llamadas concurrentes. Lo ideal sería tener una infraestructura sobredimensionada con N+1 servidores de registro.

Conclusiones

La motivación de este trabajo fue el diseño, desarrollo e implementación de una central telefónica VoIP a bajo costo, manteniendo la calidad de servicio en escenarios que abarcan la atención de grandes cantidades de usuarios. En base a lo planteado, investigado y desarrollado en este trabajo podemos arribar a las siguientes conclusiones:

- Se logró demostrar que es posible crear una infraestructura telefónica de alta disponibilidad, escalable y con calidad de servicio a partir del uso de software libre y de protocolos estándares con bajo costo y alta interoperabilidad con sistemas ya existentes.

- Si se considera que solo un porcentaje del total de clientes utiliza el servicio de telefonía concurrentemente se puede concluir que una infraestructura de estas características permite brindar el servicio a varios miles de usuarios, desde organizaciones a grandes ciudades.

- La elección de LVS (Linux Virtual Server) como balanceador de carga permitió mostrar que es posible no solamente distribuir conexiones entre los servidores de registros que se encuentran en un mismo segmento de red, sino que también permite distribuir la infraestructura a través de Internet. Esto es posible gracias al encapsulamiento IP, que además permite alivianar la carga del balanceador al no tener que procesar las respuestas desde los servidores de registro ya que estos se comunican directo con los clientes.

- Debido a la forma en que fue desarrollada la infraestructura, los servidores de registro están virtualmente expuestos a la red externa, debido al encapsulamiento IP y a la forma de balanceo. Esto requiere que se tomen los recaudos de seguridad necesarios para garantizar la estabilidad y la integridad del sistema.

- LVS permitió definir una estrategia de balanceo de carga capaz de distribuir de forma correcta y equitativa las conexiones entre los servidores de registro Asterisk, permitiendo así maximizar la cantidad de llamadas concurrentes que la central puede atender.

- A través de LVS sería posible también optimizar la comunicación entre el cliente y el servidor de registro utilizando geolocalización, re-direccionando las solicitudes del cliente al servidor de registro geográficamente más cercano a él, reduciendo de esta manera la latencia en la comunicación entre estos.

- El uso de RealTime permitió abstraer la configuración de los servidores de registro de manera que pueda ser accedida simultáneamente por cualquiera de ellos y que cualquier actualización de la misma sea inmediatamente reflejada.

- RealTime demostró ser la mejor alternativa para concentrar las configuraciones de los servidores frente a otras opciones como sistemas de archivos distribuidos o compartidos, tales como NFS (Network File Sysyem), SMB (Service Message Block o Samba) o DRBD (Distributed Replicated Block Device), ya que permite el almacenamiento de datos estructurados en una base de datos y la actualización de los mismos a través de simples sentencias SQL (Structured Query Language).

- El protocolo DUNDi permitió facilitar la localización de los usuarios dentro de la infraestructura independientemente del nodo donde estos se encuentren registrados. Permite, además, alcanzar la funcionalidad de clúster y esquema distribuido requerido evitando tener que desarrollar un sistema propio de localización de usuarios dentro de la infraestructura.

- Podemos concluir además que tanto LSV, RealTime y DUNDi son tres puntos fuertes y principales en la infraestructura, ya que su trabajo en conjunto permiten la movilidad del usuario dentro de la misma. Esto significa que los clientes no están configurados para ser registrados en un nodo específico y que pueden ser migrados tanto ante desperfectos en el servidor de registro como ante la detección de sobrecarga en el mismo.

- El uso de Heartbeat y Pacemaker resultó ser una buena elección para proveer redundancia en los servicios críticos, logrando brindar el servicio apropiado utilizando una configuración relativamente sencilla y ordenada. Además, Pacemaker resultó de utilidad para obtener monitoreo de los servicios brindados y administración de recursos en general.

- En cuanto a calidad de servicio, DiffServ es la mejor alternativa para priorizar flujos de tráfico sobre redes IP. Es un punto fundamental a considerar en telefonía IP ya que permite priorizar los flujos multimedia, disminuyendo la latencia, jitter o incluso pérdida de paquetes.

- Otro punto a tener en cuenta es mantener al dominio libre de congestión. ECN fue una manera interesante para lograr este objetivo, demostró ser una buena opción para solucionar este problema de congestión *dentro* del dominio del proveedor del servicio. Si bien ECN trae beneficios directos en datos transportados a través de TCP, los beneficios del mismo se ven reflejados en toda la red, lo cual también beneficia indirectamente al tráfico VoIP el cual es enviado en su mayoría a través de UDP.

- Para poder aplicar correctamente DiffServ y ECN es necesario tener un control absoluto del dominio donde se aplica. Lo cual es relativamente normal cuando se trata de empresas proveedoras de servicios de Internet, quienes podrían ser potenciales interesados en aplicar una tecnología como la descripta en este proyecto.

Se debe aclarar que las pruebas de calidad de servicio y telefonía IP se hicieron de forma separada, ya que por limitaciones técnicas, es decir, no contar con una infraestructura de la dimensión y la heterogeneidad del tráfico requerido, no se pudieron realizar pruebas exhaustivas aplicando todas las tecnologías investigadas.

Ya sea a limitaciones temporales y/o técnicas o a que simplemente no se encontraban dentro de los alcances de este proyecto es posible realizar implementaciones adicionales o alternativas a las planteadas hasta el momento. Algunas de estas alternativas serían:

- Utilizar otra herramienta en la capa de comunicación del clúster stack en lugar de Heartbeat para brindar alta disponibilidad. Una alternativa sería la utilización de CoroSync, que también funciona en conjunto con Pacemaker, por lo cual el cambio no debería ser demasiado radical.

- Integración de la base de datos de usuarios con LDAP. RealTime también permite integración con LDAP al igual que con base de datos relacionales, lo que permitiría integrar Asterisk a sistemas de organizaciones que estén utilizando este protocolo para almacenar información de usuarios.

- En lugar de utilizar un balanceador de carga a nivel transporte se puede optar por utilizar un proxy SIP, que funciona a nivel aplicación y se encarga de gestionar entre otras cosas las registraciones de usuarios y realizar balanceo de carga. Dos implementaciones de este tipo son OpenSips y Kamailio.

- Gracias a la abstracción de la configuración de los servidores a una base de datos relacional es posible implementar una interfaz gráfica para carga y actualización de las mismas que se adapte a las necesidades específicas de cada cliente o de quien implemente la infraestructura.

- Diseñar e implementar un esquema de seguridad general para toda la infraestructura que garantice la integridad y la confiabilidad del servicio.

Como conclusión final, el presente trabajo ha demostrado que es posible implementar una central telefónica VoIP a bajo costo basada en software libre y protocolos estándares a través de una arquitectura de alta disponibilidad capaz de brindar soporte a empresas, grandes organizaciones o a regiones provinciales. Queda abierta la invitación a que interesados puedan continuar, invertir y ampliar el desarrollo hasta aquí logrado, pudiendo tener en cuenta algunas de las recomendaciones antes mencionadas u otras que pudiesen surgir.

Bibliografía

Almquist, P. (1992). *RFC 1349 - " Type of Service in the Internet Protocol Suite".*

Andrew S.Tanenbaum, M. V. (2006). *Distributed Systems - Principles and Paradigms (2nd edition).*

Baker, F. (1995). *RFC 1812 - "Requirements for IP Version 4 Routers".*

Balliache, L. (2003). *Linux Traffic Control*. Retrieved from Differentiated Service on Linux HOWTO: http://www.opalsoft.net/qos/DS-21.htm

Braden, R. (1989). *RFC 1122 - "Requirements for Internet Hosts -- Communication Layers".*

ClusterLabs. (2010). *Pacemaker - ClusterLabs*. Retrieved from ClusterLabs: http://www.clusterlabs.org/wiki/Pacemaker

Eastlake, D. (1993). *RFC 1455 - "Physical Link Security Type of Service".*

Franco, L., & Muller, E. (2009). *IMPLEMENTACIÓN DE SERVICIOS DE VOIP UTILIZANDO ASTERISK.*

Horms Solutions. (2010). *ldirectord*. Retrieved from Horms Solutions: http://horms.net/projects/ldirectord/

Information Sciences Institute - University of Southern California. (1981). *RFC 791 - "INTERNET PROTOCOL".*

Information Sciences Institute - University of Southern California. (1981). *RFC 793 - "TRANSMISSION CONTROL PROTOCOL".*

J. Heinanen, F. B. (1999). *RFC 2597 - "Assured Forwarding PHB Group".*

J. Rosenberg, H. S. (2002). *RFC 3261 - "SIP: Session Initiation Protocol".*

K. Nichols, S. B. (1998). *RFC 2474 - "Definition of the Differentiated Services Field (DS Field)".*

K. Ramakrishnan, S. F. (2001). *RFC 3168 - "The Addition of Explicit Congestion Notification (ECN) to IP".*

Linux-HA. (2010). *Cluster Glue - Linux-HA*. Retrieved from Linux-HA: http://www.linux-ha.org/wiki/Cluster_Glue

Linux-HA. (2010). *Heartbeat - Linux-HA*. Retrieved from linux-ha.org: http://www.linux-ha.org/wiki/Heartbeat

Linux-HA. (2010). *Heartbeat Resource Agents - Linux-HA*. Retrieved from Linux-HA: http://www.linux-ha.org/wiki/Heartbeat_Resource_Agents

Linux-HA. (2010). *LSB Resource Agents - Linux-HA*. Retrieved from Linux-HA: http://www.linux-ha.org/wiki/LSB_Resource_Agents

Linux-HA. (2010). *LSB Resource Agents - Linux-HA*. Retrieved from Linux-HA: http://www.linux-ha.org/wiki/OCF_Resource_Agents

Linux-HA. (2011). *Resource Agents - Linux-HA*. Retrieved from Linux-HA: http://www.linux-ha.org/wiki/Resource_Agents

M. Spencer, B. C. (2010). *RFC 5456 - "IAX: Inter-Asterisk eXchange Version 2".*

Madsen, L. (2008). *Why cluster? An introduction to Asterisk clustering and database integration.* Retrieved from LeifMadsen Enterprises, Inc.: http://leifmadsen.com/sites/default/files/Why_Cluster_An_Introduction_to_Asterisk_Clustering_and_Database_Integration_AstriCon_2008_LMadsen.pdf

N. Spring, D. W. (2003). *RFC 3540 - "Robust Explicit Congestion Notification (ECN)".*

Perkins, C. (1996). *RFC 2003 - "IP Encapsulation within IP".*

Postel, J. (1981). *RFC 792 - "INTERNET CONTROL MESSAGE PROTOCOL".*

R. Braden, D. C. (1994). *RFC 1633 - "Integrated Services in the Internet Architecture: an Overview".*

Ramakrishnan, K. (1999). *RFC 2481 - "A Proposal to add Explicit Congestion Notification (ECN) to IP".*

Resnick, P. (2001). *RFC 2822 - "Internet Message Format".*

Rusty Russell, H. W. (2002). *Linux netfilter Hacking HOWTO*. Retrieved from Netfilter Project: http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.txt

S. Blake, D. B. (1998). *RFC 2475 - "An Architecture for Differentiated Services".*

S. Bradner, V. P. (2000). *RFC 2780 - "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers".*

S. Kent, R. A. (1998). *RFC 2401 - "Security Architecture for the Internet Protocol".*

Spencer, M. (2004). *Distributed Universal Number Discovery (DUNDi).*

T. Berners-Lee, R. F. (2005). *RFC 3986 - "Uniform Resource Identifier (URI): Generic Syntax".*

The Linux Virtual Server Project. (1998). *IPVS Software - Advanced Layer-4 Switching*. Retrieved from The Linux Virtual Server Project - Linux Server Cluster for Load Balancing: http://www.linuxvirtualserver.org/software/ipvs.html#kernel-2.6

V. Jacobson, K. N. (1999). *RFC 2598 - "An Expedited Forwarding PHB".*

voip-info.org. (2010). *Asterisk RealTime - voip-info.org*. Retrieved from voip-info.org: http://www.voip-info.org/wiki/view/Asterisk+RealTime

Apéndice A – Configuraciones de calidad de servicio

Script de configuración Routers DS sin QoS:

```
# SE BORRA LA QDISC

tc qdisc del dev $1 root

# SE CONFIGURA LA QDISK

tc qdisc add dev $1 root tbf rate 256kbit burst 16kb limit 100Kb
```

Script de configuración de Routers DS con QoS:

```
tc qdisc del dev $1 root

# Root

tc qdisc add dev $1 root handle 1:0 dsmark indices 64 set_tc_index


# Principal

tc qdisc add dev $1 parent 1:0 handle 2:0 htb

tc class add dev $1 parent 2:0 classid 2:1 htb rate 256kbit ceil 256kbit


# Clases Intermedias

tc class add dev $1 parent 2:1 classid 2:10 htb rate   128kbit ceil 256kbit prio
1

tc class add dev $1 parent 2:1 classid 2:20 htb rate    64kbit ceil 128kbit prio
2

tc class add dev $1 parent 2:1 classid 2:30 htb rate    32kbit ceil 128kbit prio
3

tc class add dev $1 parent 2:1 classid 2:40 htb rate  19.2kbit ceil 128kbit prio
4
```

```
tc class add dev $1 parent 2:1 classid 2:50 htb rate  12.8kbit ceil 128kbit prio
5


# qdisc Finales

# EF

tc qdisc add dev $1 parent 2:10 handle 10: pfifo limit 150

# AF1

tc qdisc add dev $1 parent 2:20 handle 20: gred setup DPs 3 default 2 grio

tc qdisc change dev $1 parent 2:20 gred DP 0 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 64kbit probability 0.01 prio 1

tc qdisc change dev $1 parent 2:20 gred DP 1 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 64kbit probability 0.03 prio 2

tc qdisc change dev $1 parent 2:20 gred DP 2 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 64kbit probability 0.06 prio 3

# AF2

tc qdisc add dev $1 parent 2:30 handle 30: gred setup DPs 3 default 2 grio

tc qdisc change dev $1 parent 2:30 gred DP 0 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 32kbit probability 0.01 prio 1

tc qdisc change dev $1 parent 2:30 gred DP 1 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 32kbit probability 0.03 prio 2

tc qdisc change dev $1 parent 2:30 gred DP 2 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 32kbit probability 0.06 prio 3

# AF3

tc qdisc add dev $1 parent 2:40 handle 40: gred setup DPs 3 default 2 grio

tc qdisc change dev $1 parent 2:40 gred DP 0 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.01 prio 1

tc qdisc change dev $1 parent 2:40 gred DP 1 limit 150000 min 20000 max 80000
```

```
avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.03 prio 2

tc qdisc change dev $1 parent 2:40 gred DP 2 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 19.2kbit probability 0.06 prio 3

# AF4

tc qdisc add dev $1 parent 2:50 handle 50: gred setup DPs 3 default 2 grio

tc qdisc change dev $1 parent 2:50 gred DP 0 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 12.8kbit probability 0.01 prio 1

tc qdisc change dev $1 parent 2:50 gred DP 1 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 12.8kbit probability 0.03 prio 2

tc qdisc change dev $1 parent 2:50 gred DP 2 limit 150000 min 20000 max 80000
avpkt 1000 burst 40 bandwidth 12.8kbit probability 0.06 prio 3


# Filtros

# A todos los paquetes que llegan le aplica la mascara 11111100 y corre 2 a la
derecha (saca los bits de ecn para evaluar)

tc filter add dev $1 protocol ip parent 1:0 prio 1 tcindex mask 0xfc shift 2
pass_on

# Filtro para EF

tc filter add dev $1 parent 1:0 prio 1 handle 46 tcindex classid 1:151

# Filtros para AF1

tc filter add dev $1 parent 1:0 prio 1 handle 10 tcindex classid 1:110

tc filter add dev $1 parent 1:0 prio 1 handle 12 tcindex classid 1:111

tc filter add dev $1 parent 1:0 prio 1 handle 14 tcindex classid 1:112

# Filtros para AF2

tc filter add dev $1 parent 1:0 prio 1 handle 18 tcindex classid 1:120

tc filter add dev $1 parent 1:0 prio 1 handle 20 tcindex classid 1:121
```

```
tc filter add dev $1 parent 1:0 prio 1 handle 22 tcindex classid 1:122

# Filtros para AF3

tc filter add dev $1 parent 1:0 prio 1 handle 26 tcindex classid 1:130

tc filter add dev $1 parent 1:0 prio 1 handle 28 tcindex classid 1:131

tc filter add dev $1 parent 1:0 prio 1 handle 30 tcindex classid 1:132

# Filtros para AF4

tc filter add dev $1 parent 1:0 prio 1 handle 34 tcindex classid 1:140

tc filter add dev $1 parent 1:0 prio 1 handle 36 tcindex classid 1:141

tc filter add dev $1 parent 1:0 prio 1 handle 38 tcindex classid 1:142


tc filter add dev $1 protocol all parent 2:0 prio 1 tcindex mask 0xf0 shift 4
pass_on

tc filter add dev $1 parent 2:0 prio 1 handle 1 tcindex classid 2:20

tc filter add dev $1 parent 2:0 prio 1 handle 2 tcindex classid 2:30

tc filter add dev $1 parent 2:0 prio 1 handle 3 tcindex classid 2:40

tc filter add dev $1 parent 2:0 prio 1 handle 4 tcindex classid 2:50

tc filter add dev $1 parent 2:0 prio 1 handle 5 tcindex classid 2:10
```

Configuración IPsec para el router A del escenario DS:

Contenido del archivo ipsec.conf, encargado de poner en marcha IPsec:

```
setkey -F

setkey -FP

setkey -f setkey.conf

racoon -v -F -f racoon.conf
```

Contenido del archivo setkey.conf:

```
spdadd 10.0.0.0/8 20.0.0.0/8 any -P out ipsec

esp/tunnel/10.0.0.1-20.0.0.1/require;

spdadd 20.0.0.0/8 10.0.0.0/8 any -P in ipsec

esp/tunnel/20.0.0.1-10.0.0.1/require;
```

Contenido del archive racoon.conf:

```
path pre_shared_key "sharedkey.txt";

remote 20.0.0.1

{

  exchange_mode main;

  proposal

  {

    encryption_algorithm 3des;

    hash_algorithm md5;

    authentication_method pre_shared_key;

    dh_group modp1024;

  }

}

sainfo anonymous

{

  pfs_group modp768;

  encryption_algorithm 3des;

  authentication_algorithm hmac_md5;

  compression_algorithm deflate;

}
```

Contenido del archivo sharedkey.txt:

```
20.0.0.1      1234567
```

Apéndice B – Resultados de las pruebas de calidad de servicio

Tabla de resultados Caso Base:

N° de prueba	Recibidos		
	AF11	AF12	AF13
1	181	457	263
2	415	192	269
3	453	222	232
4	199	437	240
5	445	208	231
6	134	423	295
7	278	457	140
8	314	182	388
9	310	379	184
10	127	382	397
11	473	285	311
12	203	387	309
13	358	292	311
14	195	259	169
15	137	285	198
16	190	234	376
17	226	351	245
18	222	108	288
19	393	304	368
20	182	286	194

Promedio:	**271,75**	**306,5**	**270,4**

Recibidos %	**54,35**	**61,3**	**54,08**

Perdidos %	**45,65**	**38,7**	**45,92**

N° de prueba	Tiempo en Segundos		
	AF11	AF12	AF13
1	56,43	56,37	56,21
2	56,68	56,93	56,52
3	58,63	58,59	58,46
4	56,92	56,94	56,78

5	56,72	56,59	56,24
6	54,73	54,87	54,72
7	56,43	56,44	56,24
8	57,04	57	56,9
9	56,35	56,3	56,17
10	58,43	58,46	58,24
11	56,67	56,41	56,28
12	56,84	57,19	56,6
13	56,46	56,41	56,23
14	56,4	56,44	56,42
15	56,36	56,37	56,18
16	55,31	55,46	55,27
17	56,47	56,37	56,17
18	56,88	56,61	56,35
19	61,74	61,74	61,55
20	53,81	53,84	53,65

Promedio	**56,765**	**56,7665**	**56,559**

N° de prueba	**Velocidad en Kbits**		
	AF11	**AF12**	**AF13**
1	26,4	66,7	34,5
2	60,2	27,7	39,1
3	63,5	31,2	32,6
4	28,8	63,1	34,8
5	64,5	30,2	33,8
6	20,1	63,4	44,3
7	40,5	66,6	20,5
8	45,3	26,3	56,1
9	45,2	55,4	26,9
10	17,9	53,7	56,1
11	68,6	31,3	27,6
12	26,9	55,7	44,9
13	52,1	30,3	45,5
14	44,5	35,1	48,4
15	52,8	31,4	44,2
16	46,1	26,1	56
17	39,9	51,1	37,3
18	39,8	57	30,9
19	52,3	26,1	49,2
20	48,6	32,7	46,9

Promedio	**44,2**	**43,055**	**40,48**
% Del total	34,6028888	33,7065017	31,6906095

Tabla de resultados Caso QoS1:

N° de prueba	Recibidos		
	AF11	AF12	AF13
1	497	246	97
2	496	238	124
3	491	240	107
4	493	240	102
5	492	258	105
6	497	251	95
7	495	242	99
8	497	249	94
9	497	247	105
10	495	242	106
11	496	235	124
12	496	234	112
13	498	245	102
14	494	235	117
15	496	242	106
16	498	240	108
17	495	237	114
18	495	240	113
19	497	253	95
20	496	244	105
Promedio:	**495,55**	**242,9**	**106,5**
Recibidos %	**99,11**	**48,58**	**21,3**
Perdidos %	**0,89**	**51,42**	**78,7**

n° de prueba	Tiempo en Segundos		
	AF11	AF12	AF13
1	54,96	54	60,5
2	56,16	55,81	62,54

3	55,3	55,7	63,21
4	55,03	55,81	60,34
5	57,14	55,75	59,34
6	55,31	53,67	60,41
7	55,07	53,32	60,2
8	55,35	56,7	60,35
9	55,04	62,1	49,13
10	55,2	54,88	92,32
11	55,82	55,58	67,6
12	54,93	54,83	71
13	55,36	81,63	81,72
14	55,42	73,46	73,6
15	55,07	54,82	78,97
16	55,28	70,57	70,72
17	55,02	55,05	73,68
18	55,21	55,19	80,72
19	55,27	55,04	70,1
20	55,37	73,28	73,32

Promedio	**55,3655**	**59,3595**	**68,4885**

N° de prueba	**Velocidad en Kbits**		
	AF11	**AF12**	**AF13**
1	74,4	37,5	13,2
2	72,6	35,1	16,3
3	73	29,8	13,4
4	73,7	34,3	14,8
5	70,8	38,1	14,8
6	73,9	38,5	14
7	73,9	37,3	14,7
8	73,8	35,5	14,2
9	74,3	32,7	17,6
10	73,8	36,3	6,1
11	73,1	34,8	15,1
12	74,3	35,1	13
13	74	24,7	10,3
14	73,3	26,3	13,1
15	74,1	36,2	11
16	74,1	28	12,6
17	74	35,4	12,7
18	73,7	35,8	11,5

19	74	37,8	11,1
20	73,7	27,4	11,8
Promedio	**73,625**	**33,83**	**13,065**
% Del total	61,0894457	28,0700299	10,8405244

Tabla de resultados Caso QoS2:

N° de prueba	Recibidos		
	EF	AF11	AF21
1	500	253	165
2	500	232	154
3	500	232	155
4	500	233	155
5	500	230	155
6	500	231	155
7	500	230	155
8	500	232	155
9	500	232	156
10	500	234	155
11	500	234	156
12	500	234	156
13	500	231	156
14	500	234	156
15	500	233	155
16	500	231	155
17	500	232	155
18	500	233	156
19	500	232	156
20	500	233	155
Promedio:	**500**	**233,3**	**155,8**
Recibidos %	**100**	**46,66**	**31,16**
Perdidos %	**0**	**53,34**	**68,84**

N° de prueba	Tiempo en Segundos		
	EF	AF11	AF21

1	28,07	31,4	33,9
2	25,76	32,7	31,7
3	25,85	27,43	34,42
4	25,96	35,62	36,1
5	25,48	26,91	32,83
6	25,5	26,9	33,18
7	25,42	26,75	34,48
8	25,51	26,87	26,61
9	25,38	26,88	33,3
10	25,4	26,86	34,24
11	26,09	53,01	53,33
12	26,1	86,04	86,54
13	27,66	29,15	46,4
14	26,14	36,32	29,29
15	26,16	36,74	37,122
16	26,16	27,85	29,3
17	26,1	40,44	29,21
18	26,16	41,75	42,31
19	26,16	27,85	33,66
20	26,16	42,76	43,01

Promedio	**26,061**	**35,5115**	**38,0466**

N° de prueba	Velocidad en Kbits		
	EF	AF11	AF21
1	146,5	61,4	43,2
2	159,6	60,1	45,3
3	159	69,6	37
4	158,4	53,8	35,3
5	161,4	70,3	38,8
6	161,2	70,6	38,4
7	161,6	70,7	37
8	161,2	71	47,8
9	162	71	36,3
10	161,9	71,6	37,4
11	157,6	36,3	24,1
12	157,6	22,4	14,8
13	148,7	65,2	27,6
14	157,3	53	43,8
15	157,2	52,2	34,3
16	157,2	68,2	43,5

17	157,6	47,2	43,6
18	157,2	45,9	30,3
19	157,2	68,5	38,1
20	157,2	44,8	29,6

Promedio	**157,88**	**58,69**	**36,31**

% Del total	62,4327744	23,2086365	14,3585891

Tabla de resultados Caso QoS/VoIP/Seguro:

N° de prueba	Recibidos		
	EF	AF11	AF21
1	500	239	161
2	500	232	155
3	500	232	156
4	500	233	156
5	500	232	155
6	500	232	157
7	500	232	157
8	500	232	155
9	500	232	155
10	500	231	155
11	500	232	155
12	500	232	157
13	500	232	155
14	500	232	154
15	500	234	156
16	500	232	155
17	500	233	157
18	500	231	156
19	500	232	155
20	500	233	156

Promedio	**500**	**232,5**	**155,9**

% Recibidos	**100**	**46,5**	**31,18**

% Perdidos	**0**	**53,5**	**68,82**

N° de prueba	Tiempo		
	EF	AF11	AF21
1	27,61	46,97	47,29
2	27,42	48,2	48,56
3	27,53	28,62	42,37
4	27,49	28,69	48,59
5	27,49	28,63	30,34
6	27,49	28,59	57,96
7	27,49	42,95	43,56
8	27,48	54,71	30,3
9	27,49	28,58	93,72
10	27,35	28,35	52,11
11	27,49	28,62	41,02
12	27,49	28,59	57,96
13	27,49	28,63	30,34
14	27,49	28,58	93,71
15	27,53	28,62	42,37
16	27,48	54,71	31,2
17	27,49	42,95	43,59
18	27,35	28,35	52,1
19	27,49	28,62	41,01
20	27,48	28,69	48,59

Promedio	27,481	34,5325	48,8345

N° de prueba	Velocidad		
	EF	AF11	AF21
1	148,9	41,8	28
2	150	39,6	26,2
3	149,4	66,7	30,3
4	149,6	66,8	26,4
5	149,6	66,6	42
6	149,6	66,7	22,3
7	149,6	44,4	29,6
8	149,6	34,9	42,1
9	149,6	66,8	13,6
10	150,3	67	24,5
11	149,6	66,7	31,1
12	149,6	66,7	22,3

13	149,6	66,6	43,2
14	149,6	66,8	13,6
15	149,4	66,7	30,3
16	149,6	34,9	42,1
17	149,6	44,4	29,6
18	151,4	68,6	21,3
19	149,6	66,7	31,1
20	149,6	66,8	26,4

Promedio	**149,69**	**58,81**	**28,8**

% Del total	**63,08048883**	**24,78298**	**12,13654**

Apéndice C – Configuraciones ECN

Script de configuración escenario Drop Tail:

Router 1	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root # Se crea la qdisc raiz tc qdisc add dev eth0 root tbf rate 128kbit burst 8kb limit 100Kb
ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root tbf rate 32kbit burst 2kb limit 100Kb

Router 2	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root tc qdisc add dev eth0 root tbf rate 32kbit burst 2kb limit 100Kb
ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root tbf rate 128kbit burst 8kb limit 100Kb

Script de configuración escenario RED:

Router 1	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root # Se crea la qdisc raiz tc qdisc add dev eth0 root handle 1:0 htb # Se crea una clase interna tc class add dev eth0 parent 1:0 classid 1:10 htb rate 128kbit # Se crea una qdisc hoja tc qdisc add dev eth0 parent 1:10 handle 100: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 128 probability 0.02 # Se filtra tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dst 10.1.0.0/16 flowid 1:10

ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root handle 1:0 htb # Se crea una clase interna tc class add dev eth1 parent 1:0 classid 1:20 htb rate 32kbit # Se crea una qdisc hoja tc qdisc add dev eth1 parent 1:20 handle 200: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 32 probability 0.02 # Se filtra tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 0.0.0.0/0 flowid 1:20

Router 2	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root # Se crea la qdisc raiz tc qdisc add dev eth0 root handle 1:0 htb # Se crea una clase interna tc class add dev eth0 parent 1:0 classid 1:20 htb rate 32kbit # Se crea una qdisc hoja tc qdisc add dev eth0 parent 1:20 handle 200: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 32 probability 0.02 # Se filtra tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dst 0.0.0.0/0 flowid 1:20
ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root handle 1:0 htb # Se crea una clase interna tc class add dev eth1 parent 1:0 classid 1:10 htb rate 128kbit # Se crea una qdisc hoja tc qdisc add dev eth1 parent 1:10 handle 100: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 128 probability 0.02 # Se filtra tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 10.3.0.0/16 flowid 1:10

Script de configuración escenario RED con ECN:

Router 1	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root # Se crea la qdisc raiz tc qdisc add dev eth0 root handle 1:0 htb # Se crea una clase interna tc class add dev eth0 parent 1:0 classid 1:10 htb rate 128kbit # Se crea una qdisc hoja tc qdisc add dev eth0 parent 1:10 handle 100: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 128 probability 0.02 ecn # Se filtra tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dst 10.1.0.0/16 flowid 1:10
ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root handle 1:0 htb # Se crea una clase interna tc class add dev eth1 parent 1:0 classid 1:20 htb rate 32kbit # Se crea una qdisc hoja tc qdisc add dev eth1 parent 1:20 handle 200: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 32 probability 0.02 ecn # Se filtra tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 0.0.0.0/0 flowid 1:20

Router 2	
ETH 0	# Se borra la qdisc tc qdisc del dev eth0 root # Se crea la qdisc raiz tc qdisc add dev eth0 root handle 1:0 htb # Se crea una clase interna tc class add dev eth0 parent 1:0 classid 1:20 htb rate 32kbit # Se crea una qdisc hoja tc qdisc add dev eth0 parent 1:20 handle 200: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 32 probability 0.02 ecn # Se filtra tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dst 0.0.0.0/0 flowid 1:20

ETH 1	# Se borra la qdisc tc qdisc del dev eth1 root # Se crea la qdisc raiz tc qdisc add dev eth1 root handle 1:0 htb # Se crea una clase interna tc class add dev eth1 parent 1:0 classid 1:10 htb rate 128kbit # Se crea una qdisc hoja tc qdisc add dev eth1 parent 1:10 handle 100: red limit 100000 min 10000 max 50000 avpkt 1000 burst 23 bandwidth 128 probability 0.02 ecn # Se filtra tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 10.3.0.0/16 flowid 1:10

Apéndice D – Resultados de pruebas ECN

Tabla de resultados Cubic - Drop Tail

CUBIC – DROPTAIL	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:11,9	02:12,0	02:34,1	02:20,4	02:12,6
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	724	700	740	634	624
Paquetes retransmitidos	24	36	59	27	50
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3799	3803	3272	3213	3175
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	52929	53001	53001	53001	53001
Avg owin (bytes)	33482	33852	31333	35196	31507
Máximo tiempo ocioso (ms)	9163,9	8964	25503	25628,6	25656,7
RTT Promedio (ms)	7924,7	7943	7021	8984	7429,2

CUBIC – DROPTAIL	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:29,7	02:12,9	02:10,4	02:07,1	02:33,7
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	626	636	704	691	588
Paquetes retransmitidos	50	39	24	23	28
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	2806	3235	3797	3807	2661
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	52929	52929	53001	52929	53001
Avg owin (bytes)	32525	34748	32897	37378	31340
Máximo tiempo ocioso (ms)	28473,9	24778,2	9144,8	4942,2	28755
RTT Promedio (ms)	8082,2	8560,8	8328,6	9335,2	7653,7

Tabla de resultados CUBIC - RED

CUBIC – RED	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:08,32	02:01,79	02:09,33	02:13,49	02:27,72
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	704	669	707	726	731
Paquetes retransmitidos	8	5	8	7	8
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3795	3818	3820	3783	3439
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	52929	52929	48185	52801	52929
Avg owin (bytes)	29295	28650	24370	25802	29024
Máximo tiempo ocioso (ms)	9713	11623,6	7988,7	10443,5	23785,3
RTT Promedio (ms)	6271,6	5583	4683,5	5027,1	6166,6

CUBIC – RED	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:12,27	02:12,40	02:13,35	02:11,74	02:12,37
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	727	723	732	720	724
Paquetes retransmitidos	10	4	5	7	6
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3795	3807	3817	3795	3823
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	52929	52929	52241	51481	51481
Avg owin (bytes)	26359	26043	28267	25706	26776
Máximo tiempo ocioso (ms)	9490,5	11243,6	11446	11997,3	11624,1
RTT Promedio (ms)	4893,7	5549,3	5710,3	4979,3	5552,2

Tabla de resultados CUBIC - RED con ECN

CUBIC – RED ECN	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:14,10	02:11,60	02:11,60	00:02:12,13	02:13,58
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	726	714	714	716	724
Paquetes retransmitidos	2	0	0	0	2
Segmentos máximos retransmitidos	1	0	0	0	1
Throughput (Bps)	3796	3822	3822	3822	3803
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	52929	51481	52929	52929	52929
Avg owin (bytes)	34312	35847	36305	35943	32217
Máximo tiempo ocioso (ms)	9720,8	9337,6	9341,4	9337,1	7104,9
RTT Promedio (ms)	9175,6	9435,4	9556,4	9463,5	8613,2

CUBIC – RED ECN	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:11,60	02:11,27	02:13,60	02:11,31	02:10,25
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	714	710	721	710	706
Paquetes retransmitidos	0	0	1	0	2
Segmentos máximos retransmitidos	0	0	1	0	1
Throughput (Bps)	3822	3824	3810	3823	3800
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	51481	51481	50033	50033	51481
Avg owin (bytes)	35703	34786	32457	32818	32520
Máximo tiempo ocioso (ms)	9340,1	9496,7	9245,3	9233,8	9232,5
RTT Promedio (ms)	9398,5	9155	8661,5	8644	8700,2

Tabla de resultados RENO - Drop Tail

RENO – DROP TAIL	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:08,60	02:07,46	02:07,89	02:10,77	02:11,16
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	693	689	679	710	599
Paquetes retransmitidos	7	11	8	15	37
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3803	3789	3737	3801	3149
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	30409	30409	30409	28961	52929
Avg owin (bytes)	15516	17285	15327	17556	27387
Máximo tiempo ocioso (ms)	5183,2	3970,5	5060,5	1965,3	27528,4
RTT Promedio (ms)	3849,2	4262,8	3856,2	4354,1	6503,2

RENO – DROP TAIL	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:08,26	02:05,16	02:13,99	02:07,51	02:10,66
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	686	682	704	691	704
Paquetes retransmitidos	9	12	0	20	7
Segmentos máximos retransmitidos	1	1	0	1	1
Throughput (Bps)	3813	3811	3814	3811	3804
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	27513	28961	36201	24617	31857
Avg owin (bytes)	17417	16512	25108	17623	15180
Máximo tiempo ocioso (ms)	6193,7	1983,5	9359,9	2293	5513
RTT Promedio (ms)	4347,9	3985,9	6558,9	4252,7	3792,3

Tabla de resultados RENO - RED

RENO – RED	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:07,28	02:07,67	02:10,65	02:11,94	02:10,10
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	686	688	705	710	702
Paquetes retransmitidos	3	3	3	2	4
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3811	3815	3827	3820	3797
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	37649	36201	33305	33305	39097
Avg owin (bytes)	16401	16645	17301	17263	17943
Máximo tiempo ocioso (ms)	4944,3	4129,6	6841,4	6468,3	6459,3
RTT Promedio (ms)	3980,6	4069,7	4313,4	4320,9	4442,3

RENO – RED	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:10,52	02:08,54	02:10,02	02:05,53	02:07,52
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	705	694	700	679	689
Paquetes retransmitidos	3	4	2	3	2
Segmentos máximos retransmitidos	1	1	1	1	1
Throughput (Bps)	3808	3804	3822	3824	3819
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	30409	37649	37649	36201	39097
Avg owin (bytes)	15817	15831	17829	16929	17726
Máximo tiempo ocioso (ms)	6077	5320,2	5644,1	4556,6	5703,4
RTT Promedio (ms)	3949,9	3883,6	4369,2	4108,2	4344,7

Tabla de resultados RENO – RED con ECN

RENO – RED ECN	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Duración de la transmisión (minutos)	02:10,26	02:11,33	02:13,30	02:10,81	02:10,25
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	698	704	718	700	698
Paquetes retransmitidos	0	0	2	0	0
Segmentos máximos retransmitidos	0	0	1	0	0
Throughput (Bps)	3823	3823	3811	3822	3823
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	26065	23169	27513	26353	24617
Avg owin (bytes)	18621	17156	16515	16619	17866
Máximo tiempo ocioso (ms)	5318,5	5312,4	5316,9	7220,3	5697,5
RTT Promedio (ms)	4893,4	4510,9	4403,8	4367,3	4694,6

RENO – RED ECN	Prueba 6	Prueba 7	Prueba 8	Prueba 9	Prueba 10
Duración de la transmisión (minutos)	02:08,60	02:11,58	02:11,33	02:13,51	02:10,87
Tamaño de los paquetes (Kbytes)	1000	1000	1000	1000	1000
Paquetes transmitidos	688	704	704	716	700
Paquetes retransmitidos	1	0	0	1	0
Segmentos máximos retransmitidos	1	0	0	1	0
Throughput (Bps)	3802	3823	3822	3813	3820
Min owin (bytes)	1	1	1	1	1
Max owin (bytes)	23169	24617	26065	27513	23169
Avg owin (bytes)	15730	17803	17131	16925	15288
Máximo tiempo ocioso (ms)	5320,1	5696,9	4186,5	4558,2	5320,3
RTT Promedio (ms)	4188	4678,9	4502,3	4495,4	4022,3

Tablas de promedios de los resultados

CUBIC	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	02:18,47	02:12,28	02:12,11
Tamaño de los paquetes (Kbytes)	1000	1000	1000
Paquetes transmitidos	667	716	716
Paquetes retransmitidos	36	7	1
Segmentos máximos retransmitidos	1	1	0
Throughput (Bps)	3.357	3.769	3.814
Min owin (bytes)	1	1	1
Max owin (bytes)	52.972	52.083	51.771
Avg owin (bytes)	33.426	27.029	34.291
Máximo tiempo ocioso (ms)	19.101,0	11.935,6	9.139,0
RTT Promedio (ms)	8.126,2	5.441,7	9.080,3

RENO	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	02:07,98	02:08,53	02:11,63
Tamaño de los paquetes (Kbytes)	1000	1000	1000
Paquetes transmitidos	687	693	707
Paquetes retransmitidos	9	3	1
Segmentos máximos retransmitidos	1	1	0
Throughput (Bps)	3.776	3.818	3.819
Min owin (bytes)	1	1	1
Max owin (bytes)	30.409	35.718	25.582
Avg owin (bytes)	16.043	16.782	17.431
Máximo tiempo ocioso (ms)	4.738,1	5.305,1	5.315,9
RTT Promedio (ms)	3.989,4	4.121,2	4.602,7

Tablas de desvío estandar de los resultados

CUBIC	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	00:10,26	00:06,46	00:01,25
Tamaño de los paquetes (Kbytes)	0	0	0
Paquetes transmitidos	51	19	6
Paquetes retransmitidos	13	2	1
Segmentos máximos retransmitidos	0	0	1
Throughput (Bps)	429	117	11
Min owin (bytes)	0	0	0
Max owin (bytes)	37	1.492	1.142
Avg owin (bytes)	1.949	1.670	1.647
Máximo tiempo ocioso (ms)	9.666,4	4.346,8	729,8
RTT Promedio (ms)	701,9	536,6	386,4

RENO	Droptail	RED (sin ECN)	RED (con ECN)
Duración de la transmisión (minutos)	00:02,50	00:01,98	00:01,44
Tamaño de los paquetes (Kbytes)	0	0	0
Paquetes transmitidos	31	10	9
Paquetes retransmitidos	10	1	1
Segmentos máximos retransmitidos	0	0	0
Throughput (Bps)	207	10	7
Min owin (bytes)	0	0	0
Max owin (bytes)	7.859	2.851	1.718
Avg owin (bytes)	4.228	782	999
Máximo tiempo ocioso (ms)	7.591,4	883,8	796,2
RTT Promedio (ms)	1.051,9	202,0	251,9

Apéndice E – Herramienta SIPp

Palabras claves o keywords

Palabra clave	Descripción
[service]	Utiliza el nombre de servicio especificado por el parámetro -s nombre_servicio cuando se ejecuta SIPp en línea de comando.
[remote_ip]	Dirección IP remota donde se encuentra el servidor SIP, es especificada cuando se ejecuta SIPp en línea de comando.
[remote_port]	Puerto remoto por donde se escucha el servicio. El valor por defecto es 5060, puerto SIP por defecto.
[transport]	Es el protocolo de transporte utilizado en la ejecución. Los valores posibles son TCP o UDP, siendo este último el valor por defecto.
[local_ip]	Especifica la dirección IP del UAC o UAS emulado en el escenario. Por defecto toma la dirección IP del host, pero puede ser especificado con el parámetro -i.
[local_ip_type]	Dependiendo del valor de la dirección IP, este valor puede tomar los valores 4 o 6 que corresponden a IPv4 o IPv6.
[local_port]	Es el puerto aleatorio abierto en el host para ser utilizado durante la conexión SIP.
[len]	Contabiliza la longitud total del mensaje SIP para ser utilizado en el header Content-Length.
[call_number]	Es un valor de índice que comienza con el valor 1 y se incrementa en cada llamada.
[cseq]	Genera automáticamente un número de secuencia para cada uno de los mensajes SIP.
[call_id]	Este valor identifica a cada llamada y es generado por SIPp por cada llamada nueva. Cuando se ejecuta en modo cliente este valor es obligatorio para ser utilizado en el header Call-ID.
[media_ip]	Por defecto toma el valor IP del host para ser utilizado en el RTP echo. También puede ser especificado a través del parámetro -mi.
[media_ip_type]	Toma los valores 4 o 6 dependiendo de la dirección IP del host.

[media_port]	Especifica el puerto utilizado por RTP echo. Los paquetes RTP/UDP recibidos por este puerto son repetidos a su emisor.
[auto_media_port]	Sólo para pcap.
[last_*]	El valor es reemplazado por el header en el último mensaje enviado si éste estaba presente. Ejemplos: [last_to], [last_call_id], etc. Si el header no está presente en el último mensaje enviado, este valor es descartado junto con todos los bytes hasta el fin de línea. Si el header estuvo presente en varios mensajes, estos valores se concatenan.
[field0-n file=<filename> line=<number>]	Utilizado para especificar un valor en particular de un archivo CSV. El nombre de archivo es opcional o puede especificarse con el parámetro -inf. El valor 0-n corresponde al número de campo que se encuentra dentro del archivo CSV.
[file name=<filename>]	Toma el contenido completo del archivo dentro de un mensaje.
[timestamp]	Coloca el tiempo actual en el mismo formato utilizado para los logs de error.
[last_message]	Inserta el último mensaje recibido.
[$n]	Inserta el valor de una variable cuya numeración corresponde al valor n.
[authentication]	Utilizado para especificar el header de autentificación en los mensajes SIP. El formato de este campo es el siguiente: [authentication username=nombre_usuario password=contraseña].
[pid]	Provee el ID del proceso principal de SIPp.
[routes]	Si en el elemento recv se utiliza el atributo rrs = true se almacenan todos los header Record-Route de los mensajes recibidos y pueden ser utilizados luego con esta palabra clave.
[next_url]	Si en el elemento recv se utiliza el atributo rrs = true, esta palabra clave posee todo el contenido del header Contact.
[branch]	Contiene el valor que se provee para el tag branch compuesto de la concatenación del valor obligatorio z9hG4bK + número de la llamada + índice del mensaje en el escenario
[msg_index]	Provee el índice del mensaje en el escenario.

[clock_tick]	Incluye el valor del reloj interno de SIPp en el mensaje.
[sipp_version]	Incluye el valor de la versión de SIPp en el mensaje.
[users]	Si el parámetro -users es utilizado en línea de comando, esta palabra clave contiene el número actual de usuarios instanciados en el escenario.
[userid]	Si el parámetro -users es utilizado en línea de comando, contiene el identificador del usuario actual.

Ejemplos de escenarios y archivos de datos

Escenario UAC

En el ejemplo, SIPp envía el mensaje REGISTER al servidor de registración indicado en línea de comando. Como se observará, el elemento inmediato al envío es la recepción del mensaje que indica que la registración fue realizada con éxito (200 - OK). Si del servidor se recibe cualquier otro tipo de mensaje la ejecución finalizará devolviendo un código de error. Para escenarios más complejos pueden especificarse la recepción de estos códigos indicando que estos elementos son opcionales, sin dejar de especificar la recepción de un mensaje en forma obligatoria.

```
<?xmlversion="1.0"encoding="ISO-8859-2"?>
<scenario name="registracion">
<send retrans="500">
<![CDATA[
      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      Max-Forwards: 10
      Expires: 120
      User-Agent: SIPp
      Content-Length: 0
    ]]>
</send>
<!-- Respuesta del servidor Asterisk -->
<recv response="200" />
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```

El archivo de datos debería tener la siguiente forma:

```
SEQUENTIAL
usuario;dominio.asterisk
    |          |
    |          + ->field1
    + ->field0
```

Escenario UAS

El siguiente escenario emula un Agente de Usuario Servidor, devuelve el mensaje RINGING -el teléfono está sonando- ante la llegada de una invitación de conversación por otro usuario.

En este ejemplo vemos la utilización de la palabra clave *[last_*]*, de esta manera el escenario no está sujeto a responder a un usuario en particular que inicie la llamada dado a que responderá a los headers que se encuentren en el mensaje INVITE recibido. Esto demuestra la flexibilidad de la herramienta y la posibilidad construcción de escenarios complejos sin la necesidad de fijar ciertos datos que serán utilizados durante las pruebas.

```
<?xmlversion="1.0"encoding="ISO-8859-2"?>
<scenario name="ring">
<recv request="INVITE" />
<send>
<![CDATA[
       SIP/2.0 180 Ringing
      [last_Via:]
      [last_From:]
      [last_To:];tag=[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0
    ]]>
</send>
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```

Validación de escenarios

El DTD que define la estructura de los escenarios es el siguiente:

```
<!ELEMENT CallLengthRepartition EMPTY >
<!ATTLIST CallLengthRepartition value CDATA #REQUIRED >

<!ELEMENT ResponseTimeRepartition EMPTY >
<!ATTLIST ResponseTimeRepartition value CDATA #REQUIRED >

<!ELEMENT action ( ereg+ ) >

<!ELEMENT ereg EMPTY >
<!ATTLIST ereg assign_to CDATA #REQUIRED >
<!ATTLIST ereg check_it (true|false) "false" >
<!ATTLIST ereg header NMTOKEN #IMPLIED >
<!ATTLIST ereg regexp CDATA #REQUIRED >
<!ATTLIST ereg search_in (msg|hdr) "msg" >

<!ELEMENT pause EMPTY >
<!ATTLIST pause milliseconds NMTOKEN #IMPLIED >

<!ELEMENT recv ( action? ) >
<!ATTLIST recv crlf NMTOKEN #IMPLIED >
<!ATTLIST recv optional (true|false) #IMPLIED >
<!ATTLIST recv response NMTOKEN #IMPLIED >
<!ATTLIST recv request NMTOKEN #IMPLIED >
<!ATTLIST recv lost NMTOKEN #IMPLIED >
<!ATTLIST recv rtd (true|false) #IMPLIED >
<!ATTLIST recv rrs (true|false) #IMPLIED >
<!ATTLIST recv start_rtd (true|false) #IMPLIED >

<!ELEMENT scenario ( CallLengthRepartition | ResponseTimeRepartition | pause |
recv | send )* >
<!ATTLIST scenario name CDATA #REQUIRED >

<!ELEMENT send ( #PCDATA ) >
<!ATTLIST send retrans NMTOKEN #IMPLIED >
<!ATTLIST send lost NMTOKEN #IMPLIED >
<!ATTLIST send rtd (true|false) #IMPLIED >
<!ATTLIST send start_rtd (true|false) #IMPLIED >
```