UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

APLICACIÓN DE VOZ SOBRE IP PARA UN CASO DE TELEFONÍA PÚBLICA

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN TELECOMUNICACIONES

PRESENTAN

JUAN LUIS ALMARAZ ANAYA
GUADALUPE VÁZQUEZ PÉREZ
CARLOS VILLANUEVA ZÚÑIGA

ASESOR: Ing. JESÚS SÁNCHEZ ZEPEDA

Ciudad Universitaria, mayo 2006.

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

INDICE

PÁG.

CAPÍTULO 3: SIP

INTRODUCCIÓN

El teléfono, un invento de hace más de 125 años, significó una revolución social y empresarial solamente comparable al automóvil o a la computadora personal. Hasta la aparición de la telefonía móvil no obstante, la evolución tecnológica de la telefonía en comparación con el automóvil o la computadora personal, ha sido más bien escasa, incluso la propia telefonía móvil desde el punto de vista del usuario, ha aportado mejoras funcionales más bien escasas, siendo su auténtico valor la ubicuidad. La telefonía se basa desde sus inicios en tecnologías de conmutación de circuitos que no han experimentado avances significativos, fundamentalmente por la inexistencia de impulsores a la innovación interesados en cambiar el modelo de 'voz' tradicional.

Mientras la telefonía convencional ha levantado muros a la innovación, facilidad de uso (habitualmente se conocen un 20% de las funcionalidades de las PBX y se usan menos de un 10%) y reducción de costos, los servicios a través de Internet como el e-mail y las aplicaciones Web, han revolucionado el entorno empresarial debido en gran medida a la innovación constante en aplicaciones y servicios impulsada por los propios usuarios, basada en estándares abiertos que han contribuido a su elevada difusión y a unos costos altamente competitivos, que han permitido su rápida implantación en empresas y organizaciones de todo tipo.

Hoy en día las aplicaciones en Internet manejan datos, multimedia, vídeo y música y la red se está convirtiendo en un componente esencial del negocio y de las soluciones en las empresas. Sólo en los últimos diez años, Internet y la Web han generado más innovaciones que la telefonía convencional en toda su historia, por lo que no es descabellado apuntar que la siguiente etapa de innovación en las aplicaciones y servicios Web será en el ámbito de las comunicaciones telefónicas.

En particular, para que una aplicación Web pueda interactuar de forma eficiente, integrada, flexible y económica con una conversación telefónica, esta última debe de estar codificada adecuadamente para que dicha interacción sea técnicamente posible y la tecnología que lo permite es la VozIP, que consiste en codificar la voz en paquetes de datos (sobre protocolo IP) y transportar éstos a través de redes de conmutación de paquetes. Cuando la red de transporte utilizada es Internet, hablaremos de Telefonía por Internet. Es en este último caso cuando coincidirán en el mismo medio tanto las aplicaciones Web como el servicio de telefonía, resultando de ello un potencial de innovación infinito, una auténtica revolución para la industria de desarrollo de aplicaciones y de prestación de servicios.

EL SISTEMA TELEFÓNICO ESTÁNDAR: CONMUTACIÓN DE CIRCUITOS

Los sistemas telefónicos actuales están basados por un método muy probado pero a la vez un poco ineficiente para conectar llamadas llamado: conmutación de circuitos.

Conmutación de circuitos es un concepto muy básico que ha sido usado por las redes telefónicas por más de 100 años. Cuando una llamada es establecida entre dos partes, la conexión permanece mientras dure la llamada. Debido a que estamos conectando dos puntos en ambas direcciones, esta conexión es llamada circuito, este es el fundamento de la Red Telefónica Pública Conmutada (PSTN).

A continuación se describe cómo se realiza una llamada telefónica típica:

- Levantamos el auricular y escuchamos el tono de invitación a marcar (dial tone), esto nos permite afirmar que tenemos conexión a la compañía de telefonía local.
- Marcamos el número del punto al que se desea hablar.
- La llamada es enrutada a través de la central telefónica de la operadora local de telefonía hacia el punto al que estamos llamando.

- Se establece la conexión entre nuestro teléfono y la línea del otro punto utilizando varias centrales telefónicas de interconexión a través de la ruta seguida.
- El teléfono en el otro punto suena y alguien contesta la llamada.
- La conexión abre el circuito.
- Hablamos el tiempo necesario hasta colgar el teléfono. Cuando colgamos, el circuito es cerrado, liberando nuestra línea y todas las líneas utilizadas.

Durante el tiempo de duración de la llamada el circuito es continuamente abierto entre los dos teléfonos. En los sistemas telefónicos iniciales, aproximadamente en 1960, toda llamada tenía que tener una línea dedicada del punto de inicio al punto final en toda la duración de la llamada, por ejemplo desde la Ciudad de México hacia Monterrey, las centrales entre éstos dos puntos deberían conectar piezas de alambre de cobre a lo largo de toda la trayectoria, y la llamada establecida usaría, durante el tiempo necesario, la conexión solamente para ella. Por lo tanto el costo de este tipo de llamadas era muy alto porque se tenía que pagar por todos lo kilómetros de línea de cobre utilizados por la llamada.

Las conversaciones telefónicas basadas en la actual red telefónica tradicional son más eficientes y cuestan mucho menos. Nuestra voz es digitalizada, y puede ser combinada con otras miles en un cable de fibra óptica, aunque todavía contamos con una línea de cobre que entra a nuestra casa.

Estas llamadas son transmitidas a una tasa de 64 kilobits por segundo (kbps) en cada dirección, para tener una tasa total de transmisión de 128 kbps, como existen 8 kilobits (Kb) en un kilobyte (KB), tenemos entonces una transmisión de 16 KB cada segundo en que el circuito es abierto y 960 KB cada minuto. Entonces si hablamos 10 minutos, la transmisión total será de 9600 KB, que aproximadamente es igual a 10MB, el problema es que en una conversación telefónica normal, muchos de los datos transmitidos se pierden.

Mientras estamos hablando, la otra parte esta escuchando, lo que significa que solo la mitad de la conexión está en uso en un determinado tiempo, basándonos en lo anterior, podemos decir que en realidad estamos transmitiendo 4.7 MB. Además, una cantidad de tiempo considerado en la mayoría de las conversaciones es tiempo muerto, en algunas ocasiones por algunos segundos ninguna de las partes está hablando. Si reducimos estos intervalos de silencio el archivo transmitido será más pequeño.

En lugar de enviar un continuo flujo de bytes, tanto para silencio como para datos, solamente podríamos enviar los paquetes de bytes de datos cuando éstos sean creados, esto reduciría el tiempo de uso del circuito y la cantidad de bytes transmitidos en gran medida. Esta es la base de una red telefónica de conmutación de paquetes, la cual representa una alternativa a la conmutación de circuitos, y es la vez la base de la transmisión en Voz sobre IP.

EL SISTEMA TELEFÓNICO VOIP: CONMUTACIÓN DE PAQUETES

Las redes de datos no utilizan la conmutación de circuitos. Nuestras conexiones a Internet serían más lentas si éstas mantuvieran una conexión constante con la página WEB que estuviéramos viendo en cualquier momento. En cambio, las redes de datos simplemente envían y regresan datos en la medida de lo necesario y en lugar de rutear los datos sobre líneas dedicadas, los paquetes de datos fluyen a través de los miles de caminos existentes en la red hacia el destino final. Esto es llamado conmutación de paquetes.

Mientras en la conmutación de circuitos se mantiene la conexión abierta y constante, la conmutación de paquetes abre una conexión, lo suficientemente larga para enviar un paquete de un sistema a otro. Esto se realiza como sigue:

- La computadora origen convierte los datos en pequeños paquetes, cada uno con la dirección a la que los componentes de la red tienen que enviarlos.

- Dentro de cada paquete existe el contenido, el contenido es una muestra de un e-mail, un archivo de música o cualquier tipo de archivo que será transmitido en el paquete.

La computadora origen envía el paquete a un ruteador cercano, el ruteador cercano envía el paquete a otro ruteador que está cerca de la computadora cercana. Este ruteador envía el paquete a otro cercano y así respectivamente.

Cuando la computadora destino recibe finalmente los paquetes (todos los paquetes pudieron haber tomado caminos completamente diferentes para llegar ahí) ésta utiliza las instrucciones contenidas en los paquetes para obtener los datos originales.

La conmutación de paquetes es muy eficiente, debido a que la red enruta los paquetes a través de las más baratas y menos congestionadas líneas, este tipo de conmutación es el utilizado por VoIP.

VOIP: PROTOCOLO SIP

VoIP, es un método para tomar señales de audio analógicas, del tipo que escuchamos por teléfono, y las convierte en datos digitales que pueden ser transmitidos sobre Internet.

Voz sobre IP (VoIP) es una de las tecnologías que han surgido más rápidamente en el mundo, y está revolucionando la manera en que se llevan a cabo las comunicaciones en el mundo. Como toda nueva tecnología, VoIP ha pasado por algunos problemas desde que fue utilizada por primera vez en 1995, pasando desde sus inicios como un método de comunicación de computadora a computadora, hasta ser usado con teléfonos estándar conectado a un adaptador VoIP sin la necesidad de tener una computadora conectada a Internet.

Otra área donde VoIP se ha visto envuelto es en los protocolos, interfases, usado en codificación y transmisión de paquetes de datos para voz.

A pesar de que ningún protocolo se ha declarado vencedor en VoIP, SIP (*Session Initiation Protocol*) es uno de los protocolos más desarrollados para VoIP. SIP VoIP ofrece un número de ventajas sobre otros protocolos, siendo este adoptado por el líder mundial de la industria de software Microsoft. SIP VoIP opera dentro del dominio de Internet, sin embargo actualmente los *gateways* pueden conectar un teléfono SIP hacia la PSTN en muchos países.

SIP VoIP conecta a los dos puntos involucrados, marcando a un servidor proxy y éste realiza la conexión con un usuario en una red IP. En lugar de usar un número telefónico SIP utilizará las direcciones IP para localizar al usuario al que se le quiere llamar, debido a que la mayoría de las direcciones IP son dinámicas (cambian) en lugar de estáticas, SIP puede buscar un nombre de usuario en una red IP. Una vez que se ha detectado que la otra parte ha descolgado el teléfono, se establece la conexión entre los dos usuarios o envía la llamada al buzón de voz o suena el tono de invitación a marcar si la otra parte está utilizando el teléfono.

Una de las principales características de la tecnología VoIP SIP es su portabilidad, el hardware de SIP VoIP es un simple adaptador que se conecta a un puerto Ethernet de banda ancha, por lo que los usuarios pueden tomar un teléfono en cualquier lugar del mundo y recibir llamadas, esto es porque el teléfono está conectado a Internet y no es necesario saber a que lugar pertenece, simplemente se da de alta en la red y avisa que está en línea.

Con este tipo de portabilidad, así como el ahorro en costo ofrecido hace que las conexiones de banda ancha pagadas sean más atractivas al público en general.

A continuación presentamos los diagramas en que se muestra cómo son direccionadas las llamadas en cada posible escenario de comunicación utilizando la tecnología VoIP.

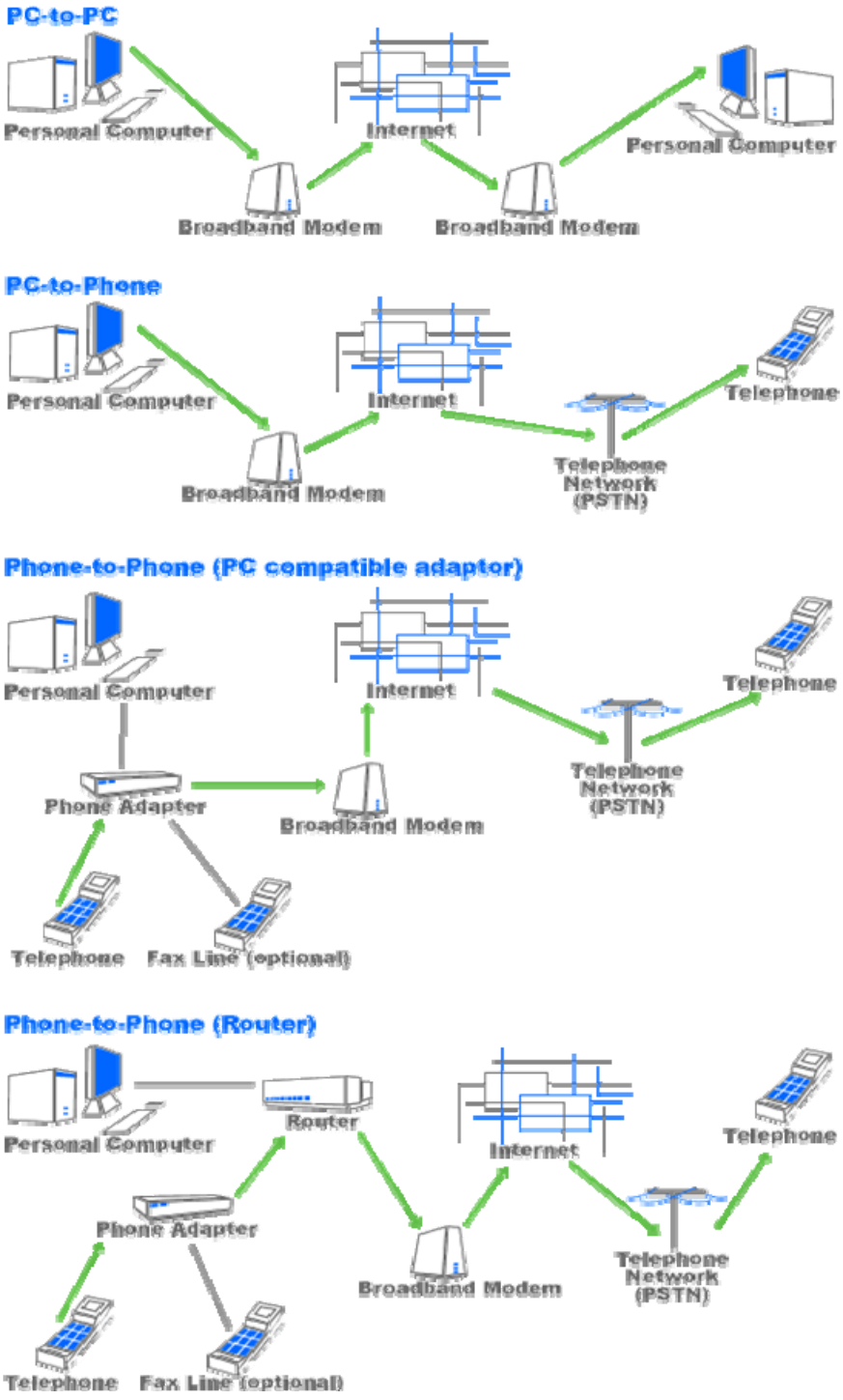


Figura A. Escenarios de Comunicación Empleando VoIP

Los sistemas de telefonía VoIP están compuestos por los siguientes elementos:

- Elementos Terminales: Éstos pueden ser teléfonos tradicionales (analógicos, GSM, ISDN), computadoras personales equipadas con audio.

- *Gateways*: Si teléfono es usado en una llamada, esta llamada necesita ser convertida al o del formato apropiado para ser transportada por Internet, esta es la tarea de los gateways.

- *Gatekeepers*/Proxies: Proveen funciones centralizadas de manejo de llamadas, proveen control de admisión de llamada, administración del ancho de banda, conversión de direcciones, autenticación, etc.

- Unidades de conferencia multipunto: Maneja conferencias multipunto.

Los componentes pueden ser implementados como hardware o software y pueden ser integrados en una sola unidad. Todos los elementos se comunican utilizando protocolos de señalización y transporte de la voz. Para garantizar la interoperabilidad entre los diferentes proveedores, se han elaborado protocolos para ambas clases de protocolos.

La tecnología de Voz sobre IP utiliza las características de la conmutación de paquetes en Internet para brindar servicio telefónico, este tipo de servicio, tiene varias ventajas sobre la conmutación de circuitos, por ejemplo: la conmutación de paquetes permite que varias llamadas telefónicas ocupen el mismo espacio que una sola en la conmutación de circuitos. Utilizando la PSTN en una llamada de 10 min. se utilizarán los 10 min. de transmisión a una tasa de 128 kbps, utilizando VoIP, la misma llamada ocupará solo 3.5 minutos de transmisión a una tasa de 64 kbps, dejando otros 64 kbps libres por esos mismos 3.5 minutos y un adicional de 128 kbps por los restantes 6.5 minutos. Podemos observar que otras 3 o 4 llamadas podrían utilizar el espacio usado por una simple llamada convencional, pero si consideramos que podemos utilizar la compresión de datos en las señales digitales de voz, se reducirá aún más el espacio de las llamadas.

Lo más interesante de VoIP es que no existe solo una manera de establecer la llamada. Existen por lo menos 3 maneras de brindar el servicio hoy en día:

- ATA - La manera más simple y más común es usar un elemento llamado ATA (Analog Telephone Adaptor). ATA nos permite conectar un teléfono estándar a una computadora o a una conexión a Internet para utilizar VoIP. ATA es un convertidor analógico- digital. Toma la señal analógica del teléfono tradicional y la convierte a una señal digital para su transmisión sobre Internet. Para iniciar el servicio simplemente se necesita conectar nuestro teléfono convencional al ATA y éste nos brindará el servicio.

- Teléfonos IP - Estos teléfonos especializados se asemejan a teléfonos normales, pero en lugar de utilizar los conectores RJ- 11, los teléfonos IP tienen un conector RJ-45 Ethernet. Los teléfonos IP conectan directamente a un router y tienen todo el hardware y software necesario en la tarjeta para manejar una llamada IP.

- Computadora a Computadora – Este es ciertamente la manera más fácil de usar VoIP, además no necesitamos pagar por llamadas de larga distancia. Existen varias compañías ofreciendo servicio gratuito o muy barato para VoIP, todo lo que necesitamos es software, un micrófono, speakers, una tarjeta de sonido y una conexión a Internet, preferentemente de banda ancha a través de un módem cable o DSL.

- Teléfono a Teléfono: A través del uso de gateways, podemos conectar directamente a cualquier teléfono estándar en el mundo. Para utilizar los beneficios que ofrecen las compañías telefónicas, debemos utilizar a uno de sus gateways para realizar la llamada, posteriormente marcar el número y la compañía nos conecta a través de su red IP.

Este último caso es básicamente el que se desarrolló en el proyecto de Tesis, conectar dos teléfonos estándar utilizando un *gateway* por medio del cual se tendrá acceso a la red IP y al servidor del protocolo SIP (especie de *gatekeepe*r) por el cual se podrá accesar a la PSTN y

completar la llamada entre los dos teléfonos estándar. En dicho *gateway* se conectarán los teléfonos públicos con conector RJ-11 y éste los enlazará utilizando el protocolo SIP de Voz sobre IP y la red de Internet, a un servidor SIP en donde se tomará el número marcado en el origen y el servidor lo entregará a una central telefónica para que la llamada sea enrutada hacia su destino final, ya sea un teléfono estándar o un teléfono celular.

A pesar de que tomará cierto tiempo, las redes de conmutación de circuitos serán reemplazadas por la tecnología de conmutación de paquetes, debido a cuestiones económicas y de requerimientos de infraestructura.

CAPÍTULO 1
REDES IP

1.1 FUNDAMENTOS Y DEFINICIONES

Las redes en general, consisten en compartir recursos, y uno de sus objetivos es hacer que todos los programas, datos y equipos estén disponibles para cualquiera de la red que así lo solicite, sin importar la localización física del recurso y del usuario. En otras palabras, el hecho de que el usuario se encuentre a 1000 kilómetros de distancia de los datos, no debe evitar que este los pueda utilizar como si fueran originados localmente.

Uno de los sucesos más importantes para la conexión en red lo constituye la aparición y la rápida difusión de la red de área local LAN (*Local Area Network*) como forma de normalizar las conexiones entre las máquinas que se utilizan como sistemas informáticos. A su nivel más elemental, una LAN no es más que un medio compartido, como un cable coaxial al que se conectan todas las computadoras y las impresoras, junto con una serie de reglas que rigen el acceso a dicho medio. La LAN más difundida, *Ethernet*, utiliza un mecanismo denominado acceso múltiple con detección de portadora y detección de colisiones (CSMA-CD - *Carrier Sense Multiple Access-Collision Detect*). Esto significa que cada equipo conectado sólo puede utilizar el medio cuando ningún otro equipo lo está utilizando. Si hay algún conflicto, el equipo que está intentando establecer la conexión la anula y efectúa un nuevo intento más adelante. La *Ethernet* más simple transfiere datos a 10Mbps, lo suficientemente rápido como para hacer inapreciable la distancia entre los diversos equipos y dar la impresión de que están conectados directamente a su destino.

Cuando se llega a un cierto punto, deja de ser práctico seguir ampliando una LAN. A veces impuesto por limitaciones físicas. Dos de los componentes importantes de cualquier red son la red telefónica y la de datos. Son enlaces para grandes distancias que amplían la LAN hasta convertirla en una red de área extensa WAN (*Wide Area Network*). Casi todos los operadores de redes nacionales ofrecen servicios para interconectar redes de computadoras, que van desde los enlaces de datos sencillos y a baja velocidad que funcionan basándose en la red pública de telefonía hasta los complejos servicios de alta velocidad adecuados para la interconexión de las LAN. Estos servicios de datos a alta velocidad suelen denominarse conexiones de banda ancha.

Internet se ha convertido en el factor más potente que guía el proceso de convergencia. Esto es debido principalmente al hecho de que la pila del Protocolo IP (*Internet Protocol*) se ha erigido como un estándar utilizado en casi cualquier servicio. La pila del Protocolo IP está compuesta principalmente por el Protocolo IP, y el Protocolo de Control del Transporte (TCP - *Transport Control Protocol*); consecuentemente el término TCP/IP refiere a la pila del protocolo al completo.

Las redes basadas en IP tienen una gran importancia en la sociedad de la información actual. A primera vista esta tecnología puede parecer un poco confusa y abrumadora pero empezaremos por presentar los componentes de red subyacentes sobre los que está construida esta tecnología.

Una red se compone de dos partes principales, los nodos y los enlaces. Un nodo es cualquier tipo de dispositivo de red como una computadora personal (PC - *Personal Computer*). Los nodos pueden comunicarse entre ellos a través de enlaces, como son los cables. Hay básicamente dos técnicas de redes diferentes para establecer comunicación entre dos nodos de una red: las técnicas de redes de conmutación de circuitos y las de redes de conmutación de paquetes. La primera es la más antigua y es la que se usa en la red telefónica y la segunda es la que se usa en las redes basadas en IP.

Una red de conmutación de circuitos crea un circuito cerrado entre dos nodos de la red para establecer una conexión. La conexión establecida está dedicada a la comunicación entre los dos nodos. Uno de los problemas inmediatos de los circuitos dedicados es la pérdida de capacidad, dado que casi ninguna transmisión usa el 100% del circuito todo el tiempo. Además, si un circuito falla en el medio de una transmisión, la conexión entera se pierde y debe establecerse una nueva.

Por otra parte las redes basadas en IP utilizan la tecnología de conmutación de paquetes, que usa la capacidad disponible de una forma mucho más eficiente y que minimiza el riesgo de posibles problemas como la desconexión. Los mensajes enviados a través de una red de conmutación de paquetes se dividen primero en paquetes que contienen la dirección de destino. Entonces, cada paquete se envía a través de la red y cada nodo intermedio o *router* de la red determina a donde va el paquete. Un paquete no necesita ser enrutado sobre los mismos nodos que los otros paquetes relacionados. De esta forma, los paquetes enviados entre dos dispositivos de red pueden ser transmitidos por diferentes rutas en el caso de que se caiga un nodo o no funcione adecuadamente.

Las soluciones de redes basadas en IP son sustitutos flexibles y económicos para soluciones que utilizan tecnologías de red antiguas. Las diversas propiedades entre estas tecnologías consisten en como se representa, gestiona y transmite la información. La información se estructura simplemente en colecciones de datos y entonces tiene sentido para la interpretación que le damos. Hay dos tipos principales de datos, analógicos y digitales y ambos poseen diferentes características y comportamientos.

Los datos analógicos se expresan como ondas continuas variables y por tanto representan valores continuos. Por otra parte los datos digitales se representan como secuencias de bits, o de unos y ceros. La digitalización permite que cualquier tipo de información sea representada y medida como datos digitales. De esta forma, el texto, sonidos e imágenes pueden representarse como una secuencia de bits. Los datos digitales pueden también comprimirse para permitir mayores tasas de transmisión y puede ser encriptada para su transmisión segura. Además una señal digital es exacta y ningún tipo de ruido relacionado puede filtrarse. Los datos digitales pueden ser transmitidos a través de tres tipos generales de medios: metal, como es el cobre, fibra óptica u ondas de radio.

1.1.1 COMPONENTES DE UNA RED DE DATOS

Una red tiene tres niveles de componentes: *software* de aplicaciones, *software* de red y *hardware* de red.

SOFTWARE DE APLICACIONES

Formado por programas informáticos que se comunican con los usuarios de la red y permiten compartir información (como archivos, gráficos o vídeos) y recursos (como impresoras o unidades de disco).

Un tipo de software de aplicaciones se denomina cliente-servidor. Las computadoras cliente envían peticiones de información o de uso de recursos a otras computadoras llamadas servidores, que controlan datos y aplicaciones. Otro tipo de software de aplicación se conoce como 'de igual a igual' (*peer to peer*). En una red de este tipo, los ordenadores se envían entre sí mensajes y peticiones directamente sin utilizar un servidor como intermediario.

SOFTWARE DE RED

Consiste en programas informáticos que establecen protocolos, para que las computadoras se comuniquen entre sí. Estos protocolos se aplican enviando y recibiendo grupos de datos con formato denominados paquetes. Los protocolos indican cómo efectuar conexiones lógicas entre las aplicaciones de la red, dirigir el movimiento de paquetes a través de la red física y minimizar las posibilidades de colisión entre paquetes enviados simultáneamente.

HARDWARE DE RED

Formado por los componentes materiales que unen las computadoras. Dos componentes importantes son los medios de transmisión que transportan las señales de las computadoras (típicamente cables o fibras ópticas) y el adaptador de red, que permite acceder al medio material que conecta a las computadoras, recibir paquetes desde el software de red y transmitir instrucciones y peticiones a otras computadoras. La información se transfiere en forma de dígitos binarios, o bits (unos y ceros), que pueden ser procesados por los circuitos electrónicos de las computadoras.

1.2 MODELO DE REFERENCIA OSI

En 1977 la Organización Internacional De Estandarización (ISO – *International Standardization Organization*) estableció un subcomité encargado de diseñar una arquitectura de comunicación. El resultado fue el Modelo de Referencia para la Interconexión de Sistemas Abiertos (OSI - *Open Systems Interconnection*), adoptado en 1983, que establece unas bases que permiten conectar sistemas abiertos para procesamiento de aplicaciones distribuidas. Se trata de un marco de referencia para definir estándares que permitan comunicar ordenadores heterogéneos.

Dicho modelo define una arquitectura de comunicación estructurada en siete niveles verticales. Cada nivel ejecuta un subconjunto de las funciones que se requieren para comunicar con el otro sistema. Para ello se apoya en los servicios que le ofrece el nivel inmediato inferior y ofrece sus servicios al nivel que está por encima de él. Idealmente, los cambios que se realicen en un nivel no deberían afectar a su nivel vecino mientras ni se modifiquen los servicios que le ofrece.

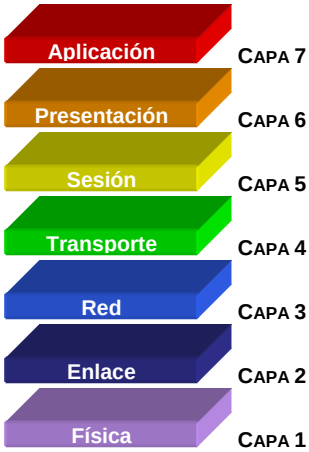


Figura 1.1. Modelo de Referencia OSI

La tarea del subcomité ISO fue definir el conjunto de niveles y los servicios proporcionados por cada nivel. Los principios aplicados para establecer un nivel fueron los siguientes:

- Diferentes niveles deben corresponder a diferentes niveles de abstracción en el manejo de los datos (por ejemplo diferencias en la morfología, la sintaxis, la semántica).
- Cada nivel debe ejecutar una función bien definida.
- Aprovechar la experiencia de protocolos anteriores. Las fronteras de niveles deben situarse donde la experiencia ha demostrado que son convenientes.
- Establecer las divisiones de los niveles de forma que se minimice el flujo de información entre ellos.
- El número de niveles debe ser suficiente para que no agrupen funciones distintas, pero no tan grande que haga la arquitectura inmanejable.
- Permitir que las modificaciones de funciones o protocolos que se realicen en un nivel no afecten a los niveles contiguos.
- Cada nivel debe interaccionar únicamente con los niveles contiguos a él (superior e inferiormente).

1.2.1 CAPA 7: APLICACIÓN

La capa de aplicación es la capa del modelo OSI más cercana al usuario, y está relacionada con las funciones de más alto nivel, proporcionando soporte a las aplicaciones o actividades del sistema, suministrando servicios de red a las aplicaciones del usuario y definiendo los protocolos usados por las aplicaciones individuales.

Es el medio por el cual los procesos las aplicaciones de usuario acceden a la comunicación por red mediante el entorno OSI, proporcionando los procedimientos precisos para ello.

Los procesos de las aplicaciones se comunican entre sí por medio de entidades de aplicación propias, estando éstas controladas por protocolos específicos de la capa de aplicación, que a su vez utilizan los servicios de la capa de presentación, situada inmediatamente debajo en el modelo.

Difiere de las demás capas debido a que no proporciona servicios a ninguna otra capa OSI, sino solamente a aplicaciones que se encuentran fuera del modelo (procesadores de texto, hojas de cálculo, navegadores web, etc.).

La capa de aplicación establece la disponibilidad de los diversos elementos que deben participar en la comunicación, sincroniza las aplicaciones que cooperan entre sí y establece acuerdos sobre los procedimientos de recuperación de errores y control de la integridad de los datos.

1.2.2 CAPA 6: PRESENTACIÓN

La capa de presentación proporciona sus servicios a la capa de aplicación, garantizando que la información que envía la capa de aplicación de un sistema pueda ser entendida y utilizada por la capa de aplicación de otro, estableciendo el contexto sintáctico del diálogo.

Su tarea principal es aislar a las capas inferiores del formato de los datos de las aplicaciones específicas, transformando los formatos particulares (ASCII, EBCDIC, etc.) en un formato común de red, entendible por todos los sistemas y apto para ser enviado por red.

Es también las responsable de la obtención y de la liberalización de la conexión de sesión cuando existan varias alternativas disponibles.

Para cumplir estas funciones, la capa de presentación realiza las siguientes operaciones:

- Traduce entre varios formatos de datos utilizando un formato común, estableciendo la sintaxis y la semántica de la información transmitida. Para ello convierte los datos desde el formato local al estándar de red y viceversa.
- Define la estructura de los datos a transmitir. Por ejemplo, en el caso de un acceso a base de datos, define el orden de transmisión y la estructura de los registros.
- Define el código a usar para representar una cadena de caracteres.
- Da formato a la información para visualizarla o imprimirla y comprimir los datos si es necesario.
- Aplica a los datos procesos criptográficos cuando es necesario.

1.2.3 CAPA 5: SESIÓN

La capa de sesión proporciona sus servicios a la capa de presentación, proporcionando el medio necesario para que las entidades de presentación de dos *hosts* que se están comunicando por red organicen y sincronicen su diálogo y procedan al intercambio de datos.

Sus principales funciones son:

- Establecer, administrar y finalizar las sesiones entre dos *hosts* que se están comunicando.
- Restaurar la sesión a partir de un punto seguro y sin pérdida de datos, en caso de una sesión falla por causa ajena al usuario, o si esto no es posible, terminar la sesión de una manera ordenada, verificando y recuperando todas sus funciones, evitando así problemas en sistemas transaccionales.
- Sincronizar el diálogo entre las capas de presentación de los dos *hosts* y administrar su intercambio de datos, estableciendo las reglas o protocolos para el dialogo entre máquinas, regulando quien habla y por cuanto tiempo.
- Conseguir una transferencia de datos eficiente y un registro de excepciones acerca de los problemas de la capa de sesión, presentación y aplicación.
- Manejar *tokens***,** objetos abstractos y únicos que se usan para controlar las acciones de los participantes en la comunicación, base de ciertos tipos de redes, como Token Ring o FDDI.
- Hacer *checkpoints*, puntos de recuerdo en la transferencia de datos, necesarios para la correcta recuperación de sesiones perdidas.

1.2.4 CAPA 4: TRANSPORTE

La capa de transporte proporciona sus servicios a la capa de sesión, efectuando la transferencia de datos entre dos entidades de sesión.

Para ello, divide los datos originados en el *host* emisor en unidades apropiadas, denominadas segmentos, que vuelve a reensamblar en el sistema del *host* receptor.

Mientras que las capas de aplicación, presentación y sesión están relacionadas con aspectos de las aplicaciones de usuario, las tres capas inferiores se encargan del transporte de datos. Además, la capa de transporte es la primera que se comunica directamente con su capa par de destino, ya que la comunicación de las capas anteriores es de tipo máquina a máquina.

La capa de transporte intenta suministrar un servicio de transporte de datos que aísle las capas superiores de los detalles del mismo, encargándose de conseguir una transferencia de datos segura y económica y un transporte confiable de datos entre los nodos de la red.

Para ello, la capa de transporte establece, mantiene y termina adecuadamente los circuitos virtuales, proporcionando un servicio confiable mediante el uso de sistemas de detección y recuperación de errores de transporte.

Se conocen con el nombre de circuitos virtuales a las conexiones que se establecen dentro de una red. En ellos no hay la necesidad de tener que elegir una ruta nueva para cada paquete, ya que cuando se inicia la conexión se determina una ruta de la fuente al destino, ruta que es usada para todo el tráfico de datos posterior.

Se pueden resumir las funciones de la capa de transporte en:

- Controlar la interacción entre procesos usuarios en las máquinas que se comunican.
- Incluir controles de integración entre usuarios de la red para prevenir perdidas o doble procesamiento de transmisiones.
- Controlar el flujo de transacciones y el direccionamiento de procesos de maquina a procesos de usuario.
- Asegurar que se reciban todos los datos y en el orden adecuado, realizando un control de extremo a extremo.
- Aceptar los datos del nivel de sesión, fragmentándolos en unidades más pequeñas aptas para el transporte confiable, llamadas segmentos, que pasa luego a la capa de red para su envío.
- Realizar funciones de control y numeración de los segmentos.
- Reensamblar los mensajes en el *host* destino, a partir de los segmentos que lo forman.
- Garantizar la transferencia de información a través de la red.

1.2.5 CAPA 3: RED

La capa de red proporciona sus servicios a la capa de transporte, siendo una capa compleja que proporciona conectividad y selección de la mejor ruta para la comunicación entre máquinas que pueden estar ubicadas en redes geográficamente distintas.

Es la responsable de las funciones de conmutación y enrutamiento de la información (direccionamiento lógico), proporcionando los procedimientos necesarios para el intercambio de datos entre el origen y el destino, por lo que es necesario que conozca la topología de la red con objeto de determinar la ruta más adecuada.

Sus principales funciones son:

- Dividir los segmentos de la capa de transporte en unidades más complejas, denominadas paquetes, a los que asigna las direcciones lógicas de los hosts que se están comunicando.
- Conocer la topología de la red y manejar el caso en que la máquina origen y la máquina destino estén en redes distintas.
- Encaminar la información a través de la red en base a las direcciones del paquete, determinando los métodos de conmutación y enrutamiento a través de dispositivos intermedios (*routers*).
- Enviar los paquetes de nodo a nodo usando un circuito virtual o datagramas.
- Ensamblar los paquetes en el *host* destino.

1.2.6 CAPA 2: ENLACE

La capa de enlace proporciona sus servicios a la capa de red, suministrando un tránsito de datos confiable a través de un enlace físico.

Se ocupa del direccionamiento físico, la topología de red, el acceso a la misma, la notificación de errores, la formación y entrega ordenada de datos y control de flujo.

Su principal misión es convertir el medio de transmisión en un medio libre de errores de cualquier tipo, realizando para ello las siguientes funciones:

- Establecer los medios necesarios para una comunicación confiable y eficiente entre dos máquinas en red.
- Agregar una secuencia especial de bits al principio y al final de los paquetes de datos, estructurando este flujo bajo un formato predefinido, denominado trama, que suele ser de unos cientos de bytes.
- Sincronizar el envío de las tramas, transfiriéndolas de una forma confiable libre de errores. Para detectar y controlar los errores se añaden bits de paridad, se usan Códigos Cíclicos Redundantes (CRC – *Cyclic Redundant Codes*) y envío de acuses de recibo positivos y negativos, y para evitar tramas repetidas se usan números de secuencia en ellas.
- Controlar la congestión de la red.
- Regular la velocidad de tráfico de datos.
- Controlar el flujo de tramas mediante protocolos que prohíben que el remitente envíe tramas sin la autorización explícita del receptor, sincronizando así su emisión y recepción.
- Encargarse del acceso de los datos al medio (soportes físicos de la red).

1.2.7 CAPA 1: FÍSICA

La misión principal de esta capa es transmitir bits por un canal de comunicación, de manera que cuanto envíe el emisor llegue sin alteración al receptor.

La capa física proporciona sus servicios a la capa de enlace de datos, definiendo las especificaciones eléctricas, mecánicas, de procedimiento y funcionales para activar, mantener y desactivar el enlace físico entre sistemas finales, relacionando la agrupación de circuitos físicos a través de los cuales los bits son transmitidos.

Sus principales funciones se resumen en:

- Definir las características materiales (componentes y conectores mecánicos) y eléctricas (niveles de tensión) que se van a usar en la transmisión de los datos por los medios físicos.
- Definir las características funcionales de la interfaz (establecimiento, mantenimiento y liberación del enlace físico).
- Transmitir el flujo de bits a través del medio.
- Manejar voltajes y pulsos eléctricos.
- Especificar cables, conectores y componentes de interfaz con el medio de transmisión, polos en un enchufe, etc.
- Garantizar la conexión (aunque no la fiabilidad de ésta).

Esta capa solamente reconoce bits individuales.

1.2.8 COMUNICACIÓN ENTRE CAPAS

Para que los paquetes de datos puedan viajar desde el origen hasta su destino, cada capa del modelo OSI en el origen debe comunicarse con su capa igual en el lugar destino. Esta forma de comunicación se conoce como “comunicaciones de par-a-par”. Las reglas y convenciones que controlan esta conversación se denominan “protocolo de la capa n”, y se ocupan del formato y significado de las unidades de datos intercambiadas.

Durante este proceso, cada protocolo de capa intercambia unidades de información entre capas iguales de las máquinas que se están comunicando, conocidas con el nombre de Unidades de Datos de Protocolo (PDU – *Protocol Data Unit*). Cada capa de comunicación, en el computador origen, se comunica con un PDU específico de capa y con su capa igual en el computador destino.

También cada capa de un modelo o arquitectura de red recibe servicios a la capa que se encuentra debajo de ella y suministra servicios a la que está por encima en la jerarquía, siendo la implantación de estos servicios transparente al usuario.

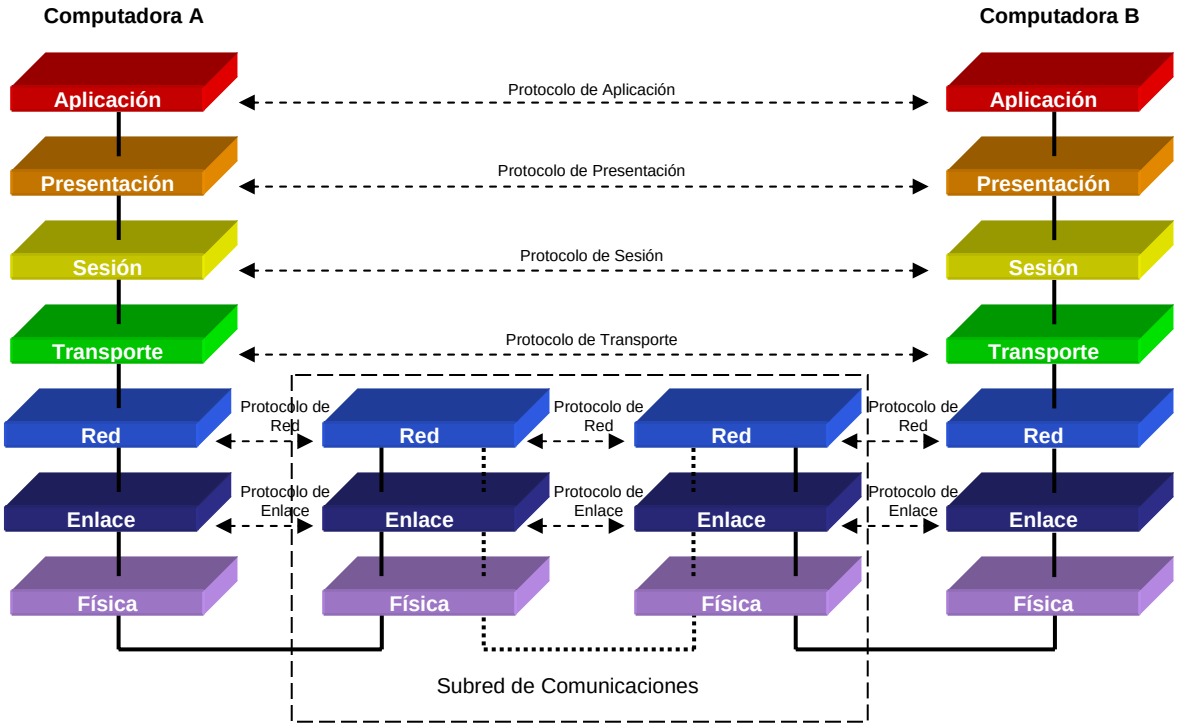


Figura 1.2. Comunicación entre Capas

En realidad, una capa de una máquina no puede transferir los datos de forma directa a su capa par de otra, si no que necesita los servicios de todas las capas que se encuentran por debajo de ella en la jerarquía de capas, pasándose la información hacia abajo hasta llegar al nivel físico, que es el que realiza el proceso de transferencia de datos.

Cada capa depende de la función de servicio de la capa OSI que se encuentra debajo de ella. Para brindar este servicio, la capa inferior utiliza el encapsulamiento para colocar la PDU de la capa superior en su campo de datos, luego le puede agregar cualquier encabezado e información final que la capa necesite para ejecutar su función. De esta forma, a medida que los datos se desplazan hacia abajo a través de las capas del modelo OSI, se agregan encabezados e información final adicionales.

1.3 MODELO TCP/IP

Aunque el modelo de referencia OSI sea universalmente reconocido, el estándar abierto de Internet desde el punto de vista histórico y técnico es el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP). El modelo de referencia TCP/IP y la pila de protocolo TCP/IP hacen que sea posible la comunicación entre dos computadoras, desde cualquier parte del mundo, a casi la velocidad de la luz.

La Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de los Estados Unidos de Norteamérica definió un conjunto de reglas que estableció cómo conectar computadoras entre sí para lograr el intercambio de información, soportando incluso desastres mayores en la subred. Fue así como se definió el conjunto de protocolos de TCP/IP.

Para los años 80 una gran cantidad de instituciones estaban interesadas en conectarse a esta red que se expandió por todo EEUU. La pila de TCP/IP consta de 4 capas principales que se han convertido en un estándar a nivel mundial.

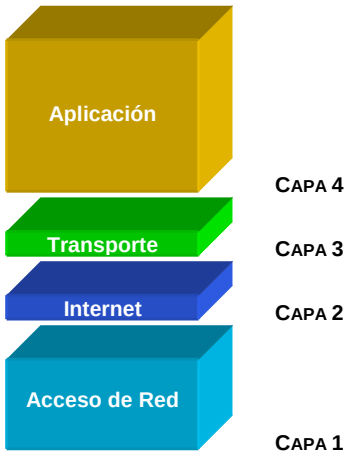


Figura 1.3. Modelo de Capas TCP/IP

1.3.1 CAPA DE APLICACIÓN

Constituye el nivel más alto del modelo TCP/IP. A diferencia del modelo OSI, se trata de un nivel simple en el que se encuentran las aplicaciones que acceden a servicios disponibles a través de Internet. Estos servicios están sustentados por una serie de protocolos que los proporcionan. Por ejemplo, el protocolo FTP (*File Transfer Protocol*), proporciona los servicios necesarios para la transferencia de ficheros entre dos computadoras.

Otro servicio, sin el cual no se concibe Internet, es el de correo electrónico, sustentado por el protocolo SMTP (*Simple Mail Transfer Protocol*).

1.3.2 CAPA DE TRANSPORTE

Esta capa proporciona una comunicación extremo a extremo entre programas de aplicación. La máquina remota recibe exactamente lo mismo que le envió la máquina origen. En esta capa, el emisor divide la información que recibe de la capa de aplicación en paquetes, le añade los datos necesarios para el control de flujo y control de errores, y se los pasa a la capa de internet junto con la dirección de destino.

En el receptor esta capa se encarga de ordenar y unir las tramas para generar de nuevo la información original.

Para implementar la capa de transporte se utilizan dos protocolos:

- UDP (*User Datagram Protocol*): Proporciona una capa de transporte no fiable de datagramas, ya que apenas añade información al paquete que envía a la capa inferior, solo la necesaria para la comunicación extremo a extremo. Lo utilizan aplicaciones como NFS y RPC, pero sobre todo se emplea en tareas de control.

- TCP: Proporciona un transporte fiable de flujo de bits entre aplicaciones. Esta pensado para poder enviar grandes cantidades de información de forma fiable, liberando al programador de aplicaciones de la dificultad de gestionar la fiabilidad de la conexión (retransmisiones, perdidas de paquete, orden en que llegan los paquetes, duplicados de paquetes, etc.) que gestiona el propio protocolo. Pero la complejidad de la gestión de la fiabilidad tiene un costo en eficiencia, ya que para llevar a cabo las gestiones anteriores se tiene que añadir bastante información a los paquetes a enviar. Debido a que los paquetes a enviar tienen un tamaño máximo, como más información añada el protocolo para su gestión, menos información que proviene de la aplicación podrá contener ese paquete. Por eso, cuando es mas importante la velocidad que la fiabilidad, se utiliza UDP, en cambio TCP asegura la recepción en destino de la información a transmitir.

1.3.3 CAPA DE INTERNET

Coloca la información que le pasa la capa de transporte en datagramas IP, le añade cabeceras necesaria para su capa y lo envía a la capa inferior. Es en esta capa donde se emplea el algoritmo de enrutamiento, al recibir un datagrama de la capa inferior decide, en función de su dirección, si debe procesarlo y pasarlo a la capa superior, o bien enrutarlo hacia otra máquina. Para implementar esta capa se utilizan los siguientes protocolos:

- *Internet Protocol* (IP): Protocolo no orientado a conexión, con mensajes de un tamaño máximo. Cada datagrama se gestiona de forma independiente, por lo que dos datagramas pueden utilizar diferentes caminos para llegar al mismo destino, provocando que lleguen en diferente orden o bien duplicados. Es un protocolo no fiable, eso quiere decir que no corrige los anteriores problemas, ni tampoco informa de ellos. Este protocolo recibe información de la capa superior y le añade información necesaria para su gestión (direcciones IP, *checksum*).

- *Internet Control Message Protocol* (ICMP): Proporciona un mecanismo de comunicación de información de control y de errores entre máquinas intermedias por las que viajaran los paquetes de datos. Esto datagramas los suelen emplear las máquinas para informarse de condiciones especiales en la red, como la existencia de una congestión, la existencia de errores y las posibles peticiones de cambios de ruta. Los mensajes de ICMP están encapsulados en datagramas IP.

- *Internet Group Management Protocol* (IGMP): Este protocolo esta íntimamente ligado a IP. Se emplea en máquinas que emplean IP *multicast*. El IP *multicast* es una variante de IP que permite emplear datagramas con múltiples destinatarios.

También en esta capa tenemos una serie de protocolos que se encargan de la resolución de direcciones:

- *Address Resolution Protocol* (ARP): Cuando una máquina desea ponerse en contacto con otra conoce su dirección IP, entonces necesita un mecanismo dinámico que permite conocer su dirección física. Entonces envía una petición ARP por *broadcast*. El protocolo establece que sólo contestara a la petición, si esta lleva su dirección IP. Por lo tanto solo contestara la máquina que corresponde a la dirección IP buscada, con un mensaje que incluya la dirección física. El software de comunicaciones debe mantener un *caché* (memoria que contiene los datos utilizados recurrentemente) con los pares IP-dirección física. De este modo la siguiente vez que hay que hacer una transmisión a es dirección IP, ya conoceremos la dirección física.

- *Reverse Address Resolution Protocol* (RARP): A veces el problema es al revés, es decir, una máquina solo conoce su dirección física, y desea conocer su dirección lógica. Esto ocurre, por ejemplo, cuando se accede a Internet con una dirección diferente, en el caso de computadoras que acceden por módem a Internet, y se le asigna una dirección diferente de las que tiene el proveedor sin utilizar. Para solucionar esto se envía una petición RARP con su dirección física, para que un servidor pueda darle su correspondencia IP.

- *Bootstrap Protocol* (BOOTP): El protocolo RARP resuelve el problema de la resolución inversa de direcciones, pero para que pueda ser mas eficiente, enviando más información que meramente la dirección IP, se ha creado el protocolo BOOTP. Este además de la dirección IP del solicitante, proporciona información adicional, facilitando la movilidad y el mantenimiento de las máquinas.

1.3.4 CAPA DE ACCESO DE RED

Esta capa se limita a recibir datagramas de la capa superior y transmitirlo al hardware de la red. Pueden usarse diversos protocolos: DLC (IEEE 802.2), Frame Relay, X.25, etc.

La interconexión de diferentes redes genera una red virtual en la que las máquinas se identifican mediante una dirección de red lógica. Sin embargo a la hora de transmitir información por un medio físico se envía y se recibe información de direcciones físicas. Un diseño eficiente implica que una dirección lógica sea independiente de una dirección física, por lo tanto es necesario un mecanismo que relacione las direcciones lógicas con las direcciones físicas. De esta forma podremos cambiar nuestra dirección lógica IP conservando el mismo hardware, del mismo modo podremos cambiar una tarjeta de red, la cual contiene una dirección física, sin tener que cambiar nuestra dirección lógica IP.

1.4 COMPARACIÓN ENTRE EL MODELO OSI Y EL MODELO TCP/IP

Los dos modelos, OSI y TCP/IP, tienen cosas en común. Constan de un número determinado de capas, y ambos utilizan protocolos independientes para comunicarse entre sí y también brindan un servicio de comunicación de extremo a extremo independiente del tipo de servicio o aplicación.

Estos modelos tienen también muchas diferencias. Una diferencia muy obvia es el número de capas con las que cuenta cada uno, el Modelo OSI tiene 7 capas y el Modelo TCP/IP cuenta solo con 4 capas.

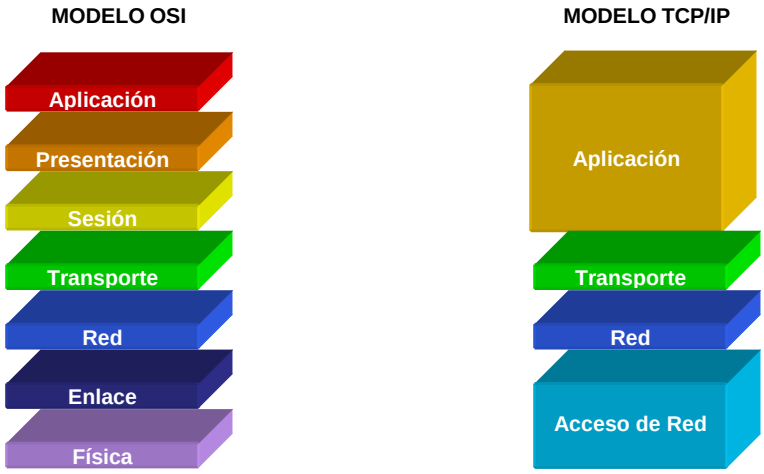


Figura 1.4. Comparación entre el Modelo OSI y el Modelo TCP/IP

Un aspecto muy importante son los conceptos utilizados en el modelo OSI. Este identifica tres conceptos principales que definen su funcionamiento:

- **Servicios**, los cuales son los procedimientos que realiza cada capa,
- **Interfaz**, que indica a los procesos de las capas superiores como acceder a ella, y
- **Protocolos**, los cuales existen para cada capa en pares, es decir, la capa de transporte del emisor y del receptor se comunican por medio del mismo protocolo. Estos protocolos pueden ser cambiados sin afectar a otras capas, siempre y cuando realicen el trabajo que le corresponde a la capa.

El modelo TCP/IP en un principio no distinguía claramente estos tres conceptos, por lo que resultaba complicado ajustar o cambiar protocolos entre capas cuando surgen nuevas tecnologías, pero fue mejorado con el tiempo.

Por su parte, el modelo OSI fue creado antes de definir los protocolos, entonces cuando se empiezan a crear redes reales, los diseñadores no tuvieron muy en claro la función específica de cada capa al definir los protocolos, y se generaron muchos problemas para hacer compatibles unas redes con otras, por lo que se tuvieron que crear subcapas para componer estos errores. Pero el modelo TCP/IP fue definido a partir de los protocolos, así que no se tenía que ajustar el modelo evitando tanto problema.

1.5 ELEMENTOS DE LA CAPA DE ACCESO A RED (TCP/IP)

A continuación, se describen los elementos constitutivos de una red IP dentro de la Capa de Acceso a Red del modelo de referencia TCP/IP, aunque también se hace referencia al modelo OSI para facilitar el entendimiento de las redes IP.

La Capa de Acceso a Red del modelo TCP/IP incluye las funciones de las Capas Física y de Enlace del modelo OSI.

1.5.1 CAPA FÍSICA (OSI)

Determina la interfaz entre el sistema y el medio de enlace, ocupándose de:

- Propiedades de la interfaz de red (topología, extensión, configuración, etc.)
- Propiedades de conexión (conector, tipo de cable, etc.)
- Propiedades eléctricas (niveles de tensión, impedancia, tipo de conductor, códigos de acceso al medio, etc.)

1.5.1.1 MEDIOS DE TRANSMISIÓN

La capa física determina el soporte físico o medio de transmisión por el cual se envían y reciben los datos. Estos medios se pueden clasificar en:

- Medios No Guiados, y
- Medios Guiados.

MEDIOS NO GUIADOS

Los rayos de más alta frecuencia y que podrían llevar más información son los X, Gamma y Ultravioleta, pero no son empleados porque son perjudiciales para los humanos, además, son difíciles de modular, por eso se usan frecuencias más bajas.

Ondas de Radio:

Una carga eléctrica en movimiento genera un campo magnético y un campo eléctrico. Si en una antena se producen corrientes eléctricas que oscilan, se producen campos eléctricos y magnéticos que se propagan, en teoría, hasta el infinito. El campo magnético rodea a la antena como una piedra arrojada al agua y el campo eléctrico es perpendicular.

Las frecuencias usadas para la transmisión de radio van más allá de los 100KHz. Las ondas de radio pierden potencial inversamente proporcional al cubo de la distancia recorrida en el aire. Pueden pasar obstáculos más fácilmente mientras menor es su frecuencia, a mayor frecuencia viajan cada vez más en línea recta y son absorbidas por la lluvia o el agua.

Todas las frecuencias sufren interferencia por campos eléctricos o magnéticos. Las ondas de radio de muy baja, baja y mediana frecuencia (10^4-10^7 Hz) viajan siguiendo la curvatura terrestre, mientras que las de alta frecuencia (10^7-10^8 Hz) pueden enviarse hacia la ionosfera en donde rebotan como si hubiera una repetidora (sin regenerar la señal) y tomar el rebote en

una retransmisora. Las frecuencias altas y muy altas son usadas para transmisiones militares.

Microondas:

Las ondas de frecuencias mayores a 100 Mhz viajan en línea recta y necesitan alinearse el transmisor y el receptor. Este tipo de señales son absorbidas por la lluvia y la tierra, por lo cual necesitan repetidoras terrestres o satélites.

La mayor parte del espectro arriba de los 100 Mhz. está estandarizado por la ITU-R, aunque hay algunas bandas que no necesitan licencia. Las bandas de 2.4 a 2.484 Ghz se usan para transmisiones médicas, científicas e industriales. Las bandas de 902 a 928 Mhz y 5.725 a 5.850 Ghz se usan para teléfonos inalámbricos y controles remotos.

Entre más alta la frecuencia, más cara es la electrónica para manejarla y más interferencia se puede tener de hornos de microondas y radares. En comparación con la fibra óptica, las microondas son más baratas porque no necesitan un cable.

Infrarrojo:

Los controles remotos trabajan con una pequeña luz infrarroja que es muy útil en las transmisiones en distancias cortas, la desventaja es que no debe haber ningún obstáculo entre el emisor y el receptor. Mientras las frecuencias de radio se acercan a las frecuencias de la luz visible se comportan menos como radio y más como luz. La luz infrarroja no se puede usar en exteriores porque el sol las anula.

Láser:

Para resolver el problema de que la brillantez del sol anula la luz infrarroja, se usan rayos láser en pequeñas distancias. El rayo láser es una luz muy potente y coherente. El rayo láser es unidireccional y para hacer LAN's se necesitan dos rayos por cada nodo.

MEDIOS GUIADOS

Par Torcido:

Se trata de pares de cobre torcidos cuya finalidad es anular la interferencia que produce cada cable, mientras más torcido esté el cable, menos interferencia habrá.

Es el medio de transmisión más barato y fácil de instalar, aunque estas características lo hacen muy versátil para muchas aplicaciones tiene también sus inconvenientes.

Existen 2 tipos de pares torcidos: el UTP (*Unshielded Twisted Pair*) que es muy susceptible al ruido generado por inducción, además la longitud puede ocasionar que actúe como antena y el STP (*Shielded Twisted Pair*), que gracias a su blindaje permite reducir el porcentaje de error, puesto que proporciona cierta inmunidad al ruido y permite extender la longitud del cable a instalar.

Coaxial:

Consiste en un conductor central de cobre cubierto de un dieléctrico, una malla de alambre y por último, el forro aislante. Es más caro que el cable de par torcido pero permite un ancho de banda más amplio para la transmisión de datos, normalmente se utiliza dos tipos de cables coaxial: de 50 Ohms para redes con señalización en banda base y 75 Ohms para señalización en banda ancha.

Fibra Óptica:

Consiste en un tubo de vidrio o plástico muy delgado a través del cual viaja información en forma de energía luminosa, es decir, la información es convertida de un formato digital a luz para ser transmitida lo que permite manejar un ancho de banda muy alto.

Debido a la naturaleza del manejo de información no se manejan medidas de resistencias ni nada relacionado con la electricidad. Por esta razón no emana campos electromagnéticos y por consiguiente no es interferido por estos campos producidos por otros equipos. Este medio es empleado principalmente para el *backbone* de las redes.

1.5.1.2 TIPOS DE COMUNICACIÓN

Las comunicaciones pueden ser clasificadas por la forma en que se realiza la transmisión en:

- ***Simplex*:** Existe un solo canal unidireccional: el origen puede transmitir al destino pero el destino no puede comunicarse con el origen. Por ejemplo, la radio y la televisión.

- ***Half-Duplex*:** Existe un solo canal que puede transmitir en los dos sentidos pero no simultáneamente, las estaciones se tienen que turnar. Esto es lo que ocurre con las emisoras de radioaficionados.

- ***Full-Duplex:*** Existen dos canales, uno para cada sentido, ambas estaciones pueden transmitir y recibir a la vez.

1.5.1.3 TOPOLOGÍAS DE RED

Las redes de computadoras surgieron como una necesidad de interconectar los diferentes *hosts* de una empresa o institución para poder así compartir recursos y equipos específicos. Pero los diferentes componentes que van a formar una red se pueden interconectar o unir de diferentes formas, siendo la forma elegida un factor fundamental que va a determinar el rendimiento y la funcionalidad de la red. La disposición de los diferentes componentes de una red se conoce con el nombre de topología de la red. La topología idónea para una red concreta va a depender de diferentes factores, como el número de máquinas a interconectar, el tipo de acceso al medio físico que se desee, etc.

Existen 3 clasificaciones de topologías:

1. La topología física, que es la disposición real de las máquinas, dispositivos de red y cableado en la red.

Bus

Esta topología tiene todos sus nodos conectados directamente a un enlace y no tiene ninguna otra conexión entre nodos. Físicamente cada *host* está conectado a un cable común, por lo que se pueden comunicar directamente, aunque la ruptura del cable hace que los *hosts* queden desconectados.

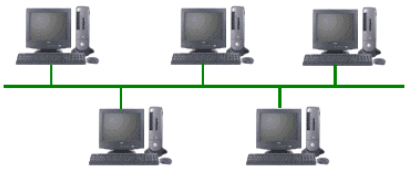


Figura 1.5. Topología de Bus

La topología de bus permite que todos los dispositivos de la red puedan ver todas las señales de todos los demás dispositivos, lo que puede ser ventajoso si desea que todos los dispositivos obtengan esta información. Sin embargo, puede representar una desventaja, ya que es común que se produzcan problemas de tráfico y colisiones, que se pueden solucionar segmentando la red en varias partes.

Anillo

Se compone de un solo anillo cerrado formado por nodos y enlaces, en el que cada nodo está conectado solamente con los dos nodos adyacentes.

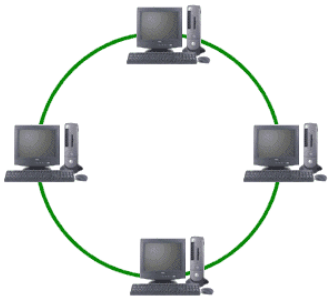


Figura 1.6. Topología de Anillo

Los dispositivos se conectan directamente entre sí por medio de cables en lo que se denomina una cadena margarita. Para que la información pueda circular, cada estación debe transferir la información a la estación adyacente.

Estrella

Tiene un nodo central desde el que se irradian todos los enlaces hacia los demás nodos. Por el nodo central, generalmente ocupado por un *hub*, pasa toda la información que circula por la red.

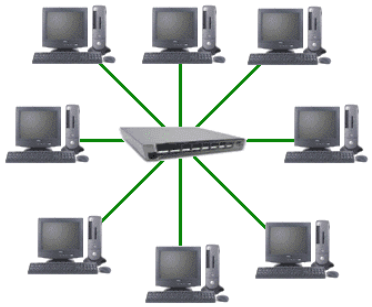


Figura 1.7. Topología de Estrella

La ventaja principal es que permite que todos los nodos se comuniquen entre sí de manera conveniente. La desventaja principal es que si el nodo central falla, toda la red se desconecta.

Estrella Extendida

La topología en estrella extendida es igual a la topología en estrella, con la diferencia de que cada nodo que se conecta con el nodo central también es el centro de otra estrella. La ventaja de esto es que el cableado es más corto y limita la cantidad de dispositivos que se deben interconectar con cualquier nodo central. La topología en estrella extendida es sumamente jerárquica, y busca que la información se mantenga local. Esta es la forma de conexión utilizada actualmente por el sistema telefónico.

Árbol

Es similar a la topología en estrella extendida, salvo en que no tiene un nodo central. En cambio, un nodo de enlace troncal desde el que se ramifican los demás nodos.

Figura 1.7. Topología en Árbol

El enlace troncal es un cable con varias capas de ramificaciones, y el flujo de

información es jerárquico. Conectado en el otro extremo al enlace troncal generalmente se encuentra un *host* servidor.

Malla Completa

Cada nodo se enlaza directamente con los demás nodos. Las ventajas son que, como cada todo se conecta físicamente a los demás, creando una conexión redundante, si algún enlace deja de funcionar la información puede circular a través de cualquier cantidad de enlaces hasta llegar a destino. Además, esta topología permite que la información circule por varias rutas a través de la red.

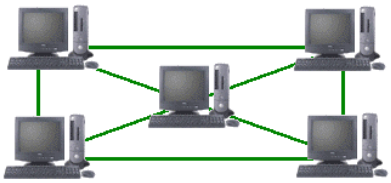


Figura 1.8. Topología en Malla Completa

La desventaja física principal es que sólo funciona con una pequeña cantidad de nodos, ya que de lo contrario la cantidad de medios necesarios para los enlaces, y la cantidad de conexiones con los enlaces se torna abrumadora.

Celular

Compuesta por áreas circulares o hexagonales, cada una de las cuales tiene un nodo individual en el centro.

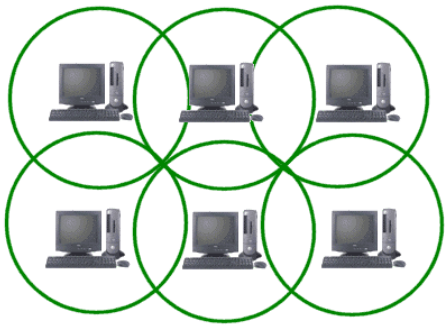


Figura 1.9. Topología de Red Celular

La topología celular es un área geográfica dividida en regiones (celdas) para los fines de la tecnología inalámbrica. En esta tecnología no existen enlaces físicos; sólo hay ondas electromagnéticas. La ventaja obvia de una topología celular es que no existe ningún medio tangible aparte de la atmósfera terrestre o el del vacío del espacio exterior. Las desventajas son que las señales se encuentran presentes en cualquier lugar de la celda y, de ese modo, pueden sufrir disturbios y violaciones de seguridad. Como norma, las topologías basadas en celdas se integran con otras topologías, ya sea que usen la atmósfera o los satélites.

Irregular

En este tipo de topología no existe un patrón obvio de enlaces y nodos. El cableado no sigue un modelo determinado; de los nodos salen cantidades variables de cables. Las redes que se encuentran en las primeras etapas de construcción, o se encuentran mal planificadas, a menudo se conectan de esta manera

2. La topología lógica, que es la forma en que las máquinas se comunican a través del medio físico. Los dos tipos más comunes de topologías lógicas son *broadcast* (*Ethernet*) y transmisión de *token*s.

   La topología de *broadcast* simplemente significa que cada *host* envía sus datos hacia todos los demás *hosts* del medio de red. Las estaciones no siguen ningún orden para utilizar la red, sino que cada máquina accede a la red para transmitir datos en el momento en que lo necesita.

   En cambio, la transmisión de *tokens* controla el acceso a la red al transmitir un *token* eléctrico de forma secuencial a cada *host*. Cuando un *host* recibe el *token* significa que puede enviar datos a través de la red. Si el *host* no tiene ningún dato para enviar, transmite el *token* hacia el siguiente *host* y el proceso se vuelve a repetir.

3. La topología matemática, mapas de nodos y enlaces, a menudo formando patrones.

Las topologías LAN más comunes son:

- *Ethernet*: topología de bus lógica y en estrella física o en estrella extendida.
- *Token Ring*: topología de anillo lógica y una topología física en estrella.
- *FDDI:* topología de anillo lógica y topología física de anillo doble.

1.5.2 CAPA DE ENLACE (OSI)

La capa de enlace proporciona un tránsito de datos confiable a través de un enlace físico. Al hacerlo, la capa de enlace de datos se ocupa del direccionamiento físico, la topología de red, el acceso a la red, la notificación de errores, entrega ordenada de tramas y control de flujo. El principal estándar usado en esta capa es el *Ethernet* (IEEE 802.3).

1.5.2.1 FUNCIONAMIENTO DE *ETHERNET* (IEEE 802.3)

Las redes *Ethernet* son actualmente las más difundidas para entornos LAN. El estándar 802.3 fue diseñado originalmente para funcionar a 10Mbps, aunque posteriormente ha sido perfeccionado para trabajar desde 100Mbps hasta 10Gbps.

Una red *Ethernet* tiene las siguientes características:

- Canal Único: Todas las estaciones comparten el mismo canal de comunicación por lo que sólo una puede utilizarlo en cada momento.
- Difusión: Debido a que todas las transmisiones llegan a todas las estaciones aunque sólo su destinatario aceptará el mensaje, el resto lo descartarán.

- Control de Acceso Distribuido: No existe una autoridad central que garantice los accesos. Es decir, no hay ninguna estación que supervise y asigne los turnos al resto de estaciones. Todas las estaciones tienen la misma prioridad para transmitir.

En las redes *Ethernet*, cuando una estación envía un mensaje a otra, no recibe ninguna confirmación de que la estación destino haya recibido su mensaje. Una estación puede estar enviando paquetes *Ethernet* a otra que está desconectada y no advertirá que los paquetes se están perdiendo. Las capas superiores son las encargadas de asegurarse que la transmisión se ha realizado de forma correcta.

El protocolo de comunicación que utilizan estas redes es el CSMA/CD. Esta técnica de control de acceso a la red ha sido normalizada constituyendo el estándar IEEE 802.3.

Cuando una estación quiere transmitir, primero escucha el canal (detección de portadora). Si está libre, transmite; pero si está ocupado, espera un tiempo y vuelve a intentarlo. Sin embargo, una vez que una estación ha decidido comenzar la transmisión puede darse el caso de que otra estación haya tomado la misma decisión, basándose en que el canal estaba libre cuando ambas lo comprobaron. Debido a los retardos de propagación en el cable, ambas señales colisionarán y no se podrá completar la transmisión de ninguna de las dos estaciones. Las estaciones que están transmitiendo lo advertirán (detección de colisiones) e interrumpirán inmediatamente la transmisión. Después esperarán un tiempo aleatorio y volverán a intentarlo. Si se produce una nueva colisión, esperarán el doble del tiempo anterior y lo intentarán de nuevo. De esta manera, se va reduciendo la probabilidad de nuevas colisiones.

Como el canal es único, todas las estaciones tienen que compartirlo. Sólo puede estar una estación transmitiendo en cada momento, sin embargo pueden estar recibiendo el mensaje más de una. La existencia de colisiones en una red no indica que exista un mal funcionamiento. Las colisiones están definidas dentro del protocolo *Ethernet* y no deben son consideradas como una situación anómala. Sin embargo, cuando se produce una colisión el canal se desaprovecha porque ninguna estación logra transmitir en ese momento. Para reducir el número de colisiones que se producen en una red se separan en grupos mediante un *switch*.

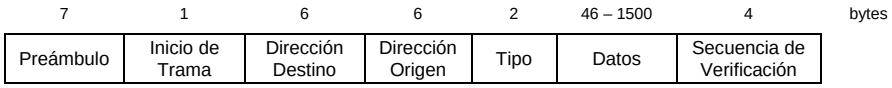
1.5.2.1.1 DIRECCIONES FÍSICAS (MAC)

Para lograr la distinción entre estaciones, se emplean las direcciones físicas también llamadas MAC (*Medium Access Control*). Todos los adaptadores de red, NIC (*Network Interface Card*), tienen asignada una dirección fija de 48 bits de fábrica. Los fabricantes garantizan que no puede haber dos tarjetas de red con la misma dirección física. Si esto llegase a ocurrir dentro de una misma red la comunicación se volvería imposible.

Los primeros 24 bits corresponden al fabricante (no puede haber dos fabricantes con el mismo identificador) y los últimos 24 bits al número de serie.

No todas las direcciones representan a máquinas aisladas, algunas de ellas se utilizan para enviar mensajes *multicast*. Esto es, enviar un mensaje a varias máquinas a la vez o a todas las máquinas de la red (*broadcast*). *Ethernet* permite que el mismo mensaje pueda ser escuchado por más de una máquina a la vez.

1.5.2.1.2 FORMATO DE LA TRAMA



- Preámbulo (56 bits): Patrón de unos y ceros que indica a las estaciones receptoras que una trama es *Ethernet*. Cuando dicho patrón está codificado en Manchester diferencial, se obtiene una señal cuadrada, empleada por el receptor para sincronizarse con el reloj del emisor.
- Inicio de trama (8 bits): Delimitador que finaliza con dos bits 1 consecutivos, y que sirve para sincronizar las porciones de recepción de trama de todas las estaciones de la red.
- Dirección Destino (48 bits): Contiene la dirección del destinatario, que puede ser de *unicast*, en la cual se codifica la dirección MAC del destinatario, *multicast*, en la cual se codifica la dirección de un grupo o *broadcast*.
- Dirección Origen (48 bits): Contiene la dirección MAC del dispositivo que originó la trama.
- Tipo (16 bits): Especifica el protocolo de capa superior que recibe los datos una vez que se ha completado el procesamiento *Ethernet*.
- Datos (46 – 1500 bytes): Incluye los datos enviados en la trama. Debido a la especificación IEEE 802.3 indica que una trama no puede tener un tamaño inferior a 64 bytes, se emplea también para completar dicha longitud.
- Secuencia de Verificación (32 bits): Contiene un valor de verificación CRC, creado por el dispositivo emisor y recalculado por el dispositivo receptor para verificar la existencia de tramas dañadas.

1.5.2.1.3 ELEMENTOS DE INTERCONEXIÓN

***Hub*:** Punto central al que se conectan los dispositivos de una red *Ethernet* para comunicarse entre sí, brindando una topología física de estrella. Su funcionamiento interno es parecido al de un conmutador eléctrico en el que conectamos varias clavijas al mismo tiempo con una sola toma, proporcionando una topología lógica de bus. El problema del *hub* es que comparte todo el tráfico de red entre los miembros de esta lo cual puede llegar a saturarla si existen muchas comunicaciones simultáneas.

***Bridge*:** Se utiliza para segmentar grandes redes en redes más pequeñas. De esta forma sólo saldrá de la red pequeña el tráfico destinado a otra red pequeña diferente mientras que todo el tráfico interno seguirá en la misma red. Con esto se consigue una reducción del tráfico de red.

***Switch*:** Es una especie de *hub* mejorado, puesto que no difunde las tramas *Ethernet* por todos los puertos, sino que las retransmite sólo por los puertos necesarios. Cada puerto tiene un *buffer* para almacenar tramas *Ethernet*. Además, puede trabajar a diferentes velocidades en sus ramas.

Para realizar su trabajo, los *switches* poseen una tabla dinámica de direcciones físicas y números de puerto. Su procesador analiza las tramas entrantes y busca la dirección física en su tabla. Si la encuentra, simplemente

la reenvía por el puerto indicado. Si por el contrario, no la encuentra, entonces la difunde por todas sus ramas.

1.5.2.1.4 DOMINIOS DE COLISIÓN

Un dominio de colisión es un segmento del cableado que comparte colisiones. Cada vez que se produzca una colisión dentro de un mismo dominio de red, afectará a todos los dispositivos conectados a ese dominio pero no a los pertenecientes a otros dominios de colisión.

Todas las ramas de un *hub* forman un mismo dominio de colisión. Por el contrario, cada rama de los *switches* constituye un dominio de colisión, motivo por el cual la utilización de *switches* reduce el número de colisiones y mejora la eficiencia de las redes.

1.6 ELEMENTOS DE LA CAPA INTERNET

1.6.1 PROTOCOLO IP

Los estándares MIL-STD-1777 y RFC-0791 determinan el protocolo IPv4 para la interconexión de redes IP. En Internet se conectan redes individuales mediante *routers*. El *router* realiza la operación de enrutar o determinar la ruta que debe seguir un paquete, mediante el uso de las direcciones IP para llegar a su destino.

1.6.1.1 DIRECCIONES IP

El sistema de direcciones de Internet, estableció un sistema jerárquico de direcciones, asignándose un número a cada red y un número a cada dispositivo particular dentro de la red. El sistema estableció que las direcciones estarían contenidas en una etiqueta de 32 bits. Resulta así que cada máquina tiene una única dirección, que en binario tendría un aspecto como:

11001100.01111011.00000010.01001011

La dirección IP es lo más parecido al número de teléfono de los *hosts*. La primera parte de una dirección IP identifica la red a la cual pertenece el *host*, mientras que la segunda parte identifica al *host* dentro de la red. El número de red también es conocido como prefijo de red y como es de esperarse, todos los *hosts* dentro de una red comparten el mimo prefijo.

Para que sean más fáciles de entender, suelen representarse en números decimales mediante un conjunto de 4 cantidades separadas por puntos: 204.123.2.75. Por esta razón este tipo de notación se conoce como Notación Decimal. Por supuesto cada conjunto de cifras está comprendido en el rango entre 0 y 255. Las cifras de la izquierda son las más significativas.

Naturalmente, todo esto necesita una supervisión y control. Originalmente, era la NSF (*National Science Foundation*), la encargada de gestionar lo relativo a la organización de Internet, después delegó en *Network Solutions*, una compañía USA; esta colaboración recibió el nombre genérico de *InterNIC*, por último, a partir de 1998 es ICANN, una organización internacional sin ánimo de lucro.

1.6.1.1.1 CLASES DE DIRECCIONES IP (*CLASSFULL ADDRESSING*)

Para poder soportar diferentes tamaños de red, los diseñadores del protocolo IP inicialmente decidieron dividir las direcciones IP en clases, donde cada clase delimita el número de *hosts* que puede tener cada red, mediante el tamaño del prefijo.

Figura 1.10. Clases de Direcciones IP

Las clases anteriores se usan de la siguiente forma:

- Clase A: Pocas redes, cada una con muchos *hosts* (16'777,216).
- Clase B: Un número medio de redes, cada una con un número medio de *hosts* (65,536).
- Clase C: Muchas redes, cada una con pocos *hosts* (256).
- Clase D: Permite hacer *multicasting*.
- Clase E: Reservado para el futuro.

1.6.1.1.2 DIRECCIONAMIENTO IP SIN CLASE (*CLASSLESS ADDRESSING*)

Debido a la naturaleza del esquema de direccionamiento *classfull* se hace un uso ineficaz del limitado número de direcciones IP. Por ejemplo, algunas empresas pequeñas, solicitaban direcciones IP clase C, teniendo como máximo 256 *hosts*. En el caso de requerir una red un poco mayor, tendrían que emplearse varias direcciones IP clase C. Por otro lado, algunas empresas medianas y grandes, solicitaban direcciones IP clase B, pudiendo alojar hasta 65,535 *hosts*, subutilizándose su capacidad. Sucedía algo peor cuando se empleaban direcciones IP clase A.

Conforme se fue agotando el espacio de direccionamiento IP ante el crecimiento desbordado de las redes IP y sobre todo de Internet, se ideó un esquema para hacer más eficiente la asignación de direcciones IP, consistente en un esquema en el que no hay clases de red, sino que se usan redes con un tamaño de prefijo que puede ser diferente a los del esquema *classfull*.

El número de máscara determina el tamaño de prefijo de cada red. Dicha máscara tiene un formato similar a las direcciones IP, por ejemplo, el número de máscara 255.255.224.0, significa que la red tiene un prefijo de *host* de 19 bits, es decir igual al número de unos. Con esta máscara se pueden alojar hasta 8192 *hosts.*

1.6.1.2 FORMATO DE LA TRAMA DE IPV4

El formato de la trama (denominado datagrama) contiene los campos que se muestran a continuación para el protocolo IPv4. El protocolo IP no tiene previsto el control de flujo, la protección de la secuencia de datos u otros servicios ofrecidos del tipo *host* a *host*.

1.6.1.2.1 CAMPOS DE INFORMACIÓN GENERAL

- VER (4 bits): Indica la versión del protocolo IP.
- LON (4 bits): Indica la longitud del encabezado en unidades de 4 bytes. El valor por omisión es 5 (que corresponde a 20 bytes sin campo opcional).
- Tipo Servicio (8 bits): Contiene información que permite gestionar la calidad del servicio dentro de una red IP.
  - Prioridad (3 bits): Es una medida de la naturaleza y prioridad de este datagrama: rutina, prioridad, inmediato, *flash*, *flash override,* crítico, control de red interna, control de red.
  - D (1 bit): Retardo admitido sobre el datagrama.
  - T (1 bit): Conectividad admitida
  - R (1 bit): Probabilidad de descarte
  - MBZ (1 bit): Reservado para uso futuro

1.6.1.2.2 CAMPOS DE INFORMACIÓN PARA LA SEGMENTACIÓN

- Longitud Total (16 bits): Indica la longitud total del datagrama, incluyendo el encabezado, en unidades de bytes. Por razones de fragmentación, se usa un valor muy inferior al máximo permitido.
- Identificación (16 bits): Identificador de dirección origen, destino y protocolo de usuario.
- Flags (3 bits): Empleado para informar de la segmentación y facilitar el ensamble de datagramas en el destino.
  - Reservado (1 bit): No tiene aplicación
  - *Do not Fragment* (1 bit): Indica si se permite o no la fragmentación de datagramas:
  - *More Fragments* (1 bit): Indica si el datagrama es único o existen más fragmentos.
- Desplazamiento (13 bits): Indica la posición del datagrama en el mensaje original en unidades de 8 *bytes*. En los segmentos sucesivos, señala la cantidad de *bytes* transmitidos hasta el momento
- Tiempo (8 bits): Especifica el número de segundos que se permite al Datagrama circular por la red antes de ser descartado.
- Protocolo (8 bits): Indica el protocolo de capa superior que transporta, actúa como dirección de servicio SAP (*Service Access Point*). Puede tomas valores decimales del 0 al 255.

- *Checksum* Encabezamiento (16 bits): Sirve para realizar detección de errores en el encabezado. Éste campo cambia en cada salto, puesto que los campos Tiempo y *Flags* cambian.

1.6.1.2.3 CAMPOS DE DIRECCIONES Y OPCIONAL

- Dirección IP Fuente (32 bits): Dirección de origen del datagrama.
- Dirección IP Destino (32 bits): Dirección de destino del datagrama.
- Opciones (M bits): Información adicional.
- Relleno (N bits): Completa el número de *bytes* mínimos

1.6.2 FUNCIONAMIENTO DEL PROTOCOLO IP

La operación de redes como X.25 o *Frame Relay* es en el modo orientado a conexión, mientras que una red IP basa su funcionamiento en modo no orientado a conexión. La operación no orientada a conexión requiere del análisis del nivel de protocolo IP, en tanto en la operación orientada a conexión, una vez establecido el circuito virtual no requiere dicho tratamiento.

El datagrama puede ser enrutado mediante una tabla de ruta estática, la cual solo dispone de alternativas en caso de que un enlace no tenga disponibilidad, o dinámica, la cual responde a errores, congestión de la red, costo y retardo. Puede soportar también requerimientos de seguridad y prioridad de datagramas. Estos aspectos pueden demorar a un datagrama en la red, consumiendo *buffer* de memoria y capacidad de transporte e incluso puede mantenerlo en un ciclo indefinido dentro de los *routers*.

Los principales elementos funcionales del protocolo IP son:

Tiempo de Vida (TTL): Empleado para reducir el riesgo de demora o pérdida del datagrama. Indica el tiempo que el datagrama puede permanecer en Internet. Como el datagrama consume memoria y recursos en la red, al expirar este tiempo, se descarta. Cada ingreso de un datagrama a un *router* le descuenta una unidad de TTL; además en tanto el datagrama permanece en el *buffer* de un nodo se descuenta el valor de TTL en una unidad por segundo.

El protocolo de la capa de transporte destino debe detectar la ausencia de un datagrama correlativo mediante la información Longitud Total (TL), *Flag* y Desplazamiento (FO), con lo que se interrumpe el ensamble. Un *router* intermedio no está autorizado a efectuar ensamble ya que no posee la seguridad que los distintos segmentos pasan por dicho *router*. El protocolo IP dispone del protocolo ICMP para enviar reportes cuando se produce un descarte ya sea por: final del tiempo de vida, congestión de la red y errores de bits en el encabezado.

Calidad del Servicio (ToS): En IP se tiene la oportunidad de definir la calidad del servicio en los siguientes términos:

- Procedencia (importancia relativa del datagrama, 8 niveles),
- Retardo admitido del datagrama (2 niveles),
- Importancia para la seguridad del datagrama (2 niveles).

Ensamble de Datos Fragmentados: Para este funcionamiento se utilizan los campos TL, FO y *Flag* MF que indica si se trata del último datagrama.

IP es responsable por la ruta de los datos pero no lo es por la integridad de los mismos. No permite el secuenciamiento, control de flujo, apertura y cierre de la conexión y reconocimiento del servicio. TCP se encarga de reconocer cuando un datagrama se ha perdido en Internet.

1.6.3 PROTOCOLOS DE ENRUTAMIENTO

Se entiende por enrutamiento al proceso que permite determinar la mejor ruta para que un paquete llegue a su destino. Se puede efectuar mediante *switches* o *routers* de acuerdo con el tipo de redes involucradas. El *switch* se prefiere por el mínimo retardo, su bajo costo, sus pocas conexiones y mínimo planeamiento. Mientras que los *routers* se usan para interconectar redes LAN y WAN, requiriendo un mayor grado de configuración para interpretar correctamente la información de enrutamiento.

El *switch* y *router* son elementos que aprenden de la red, analizan la dirección de cada paquete y pueden formar una tabla de enrutamiento que les permite determinar el camino que debe seguir cada datagrama para llegar a su destino.

El *router* tiene la capacidad de optimizar el camino del paquete de datos analizado el costo, retardo de tránsito, congestión de red y distancia medida en el número de *routers* que el datagrama debe atravesar en el trayecto. La tabla de enrutamiento contiene sólo el próximo paso en la red.

La mayor jerarquía de enrutamiento es el Sistema Autónomo (AS - *Autonomous System*) que es una colección de redes bajo una administración de dominio común. Los protocolos de resolución interior (IGP - *Interior Gateway Protocol*) permiten la comunicación interna del AS, en tanto que, los de resolución exterior (EGP - *Exterior Gateway Protocol*) lo hacen entre dominios distintos. Así un AS es un grupo de *routers* que utilizan un mismo protocolo de enrutamiento. Cada AS puede ser dividido en un número de áreas; un *router* con múltiples interfaces puede participar de múltiples áreas. Estos *routers* se denominan *routers* de borde y mantienen separadas las bases de datos topológicas de cada área.

Los *routers* emplean el principio de Mínima Acción para determinar un camino para el mensaje. Esto significa que los *routers* escogen las rutas basándose en diversos parámetros llamados métricas, de tal manera que eligen las rutas más convenientes de acuerdo a esas métricas, por ejemplo, el menor número de saltos.

La métrica es un estándar de medida que permite efectuar las operaciones de enrutamiento. Cada protocolo utiliza diferentes métricas o un conjunto de ellas para seleccionar la mejor ruta.

La tabla de enrutamiento es la responsable del camino del paquete en la red. Esta tabla realiza un mapa de la topología de la red para determinar el próximo paso hacia el destino final. La métrica para una ruta particular es el agregado de varias características asignadas a un enlace. Existen diversos tipos de métricas, algunos protocolos de enrutamiento utilizan sólo una de ellas mientras que otros usan varias alternativas. Algunas posibles métricas de los protocolos de enrutamiento son las siguientes:

- Calidad del Enlace: Existencia de errores en el trayecto.
- Longitud del Trayecto: Número de saltos o *routers* intermedios en la red.
- Retardo de Tránsito: Tiempo de propagación que se determina mediante un *Ping* de ICMP.

- Ancho de Banda del Enlace: Capacidad de tráfico disponible entre *routers*.
- Disponibilidad: Grado de ocupación del CPU del *router*.
- Costo: Toma en cuenta el valor de conexión de la ruta.

1.6.3.1 CLASIFICACIÓN DE LOS PROTOCOLOS DE ENRUTAMIENTO

Los principales criterios para la clasificación de los protocolos de enrutamiento son:

- **Estático/Dinámico:** Se refiere a la posibilidad de fijar una ruta determinada en el caso estático (*Route of Last Resort*) o enrutamiento variable para una adaptación en tiempo real.
- **Simple/Múltiple:** Referido a la posibilidad de permitir la multiplexación por varias líneas. Cuando es posible el trayecto múltiple, las vías para distribuir los paquetes son dos:
  - Balance por paquete, en el cual los paquetes son distribuidos por rutas de igual métrica, y
  - Balance por destino, en el cual se asignan rutas por cada nuevo destino. Con esto se tiende a preservar el orden de los paquetes. TCP acomoda en orden los paquetes pero puede degradarse el desempeño. Si bien IP es un protocolo no orientado a conexión los *router* preservan en lo posible la ruta.
- **Plano/Jerárquico:** En la topología plana todos los *routers* tienen igual jerarquía; en la jerárquica los *routers* forman un *backbone* para el tráfico principal.
- **Dominio:** Los protocolos de enrutamiento son distintos si trabajan en el mismo dominio (intra-dominio) o entre dominios (inter-dominio).
- **Algoritmo:** Se disponen de dos tipos de algoritmos para obtener la tablas de enrutamiento:
  - ***Distance Vector:*** Desarrollado por Bellman-Ford que requiere de datos sobre el número de saltos y el costo para definir las rutas. El costo puede indicar un valor en $/min o bien una ponderación debida al retardo: por ejemplo un peso de 1 para 128 kb/s y de 10 para 64 kb/s.
  - ***Dijkstra Algorithm:*** Es un protocolo LSA (*Link State Advertisements*) que acumula información acerca de las rutas y enlaces de la red, verificando su estado en cada momento, de tal modo que cuando ocurre algún cambio en la topología de la red, las tablas de enrutamiento se actualizan inmediatamente. Esta información es usada por el algoritmo para reconocer el camino mas corto a cada nodo.

1.6.3.2 PROTOCOLOS DE ENRUTAMIENTO DE CAPA 2

Para que los *routers* se puedan comunicar con los *switches* o segmentos de red que están conectados a él, necesitan un conjunto de protocolos y procesos. El más simple de los procesos que permiten definir direcciones y rutas es el de resolución de direcciones MAC en una red LAN es realizado por alguno de los protocolos siguientes:

ARP: Protocolo usado para anunciarse mediante direcciones de capa IP. Permite comunicarse con un usuario IP sin conocer la dirección MAC del mismo. ARP emite un mensaje ARP *REQUEST* para solicitar la dirección MAC correspondiente a una dirección IP para todas las unidades MAC en la red LAN. La respuesta es un mensaje ARP *RESPONSE* con la dirección MAC que corresponde a la dirección IP indicada.

Un datagrama ARP es un paquete de corta longitud que contiene los siguientes campos:

- Identificador de Tipo Hardware (16 bits): Tipo de dispositivo de red que envía el paquete.
- Identificador de Protocolo (16 bits): Protocolo de red usado por el dispositivo de red.
- HLEN/PLEN (16 bits): indica la longitud de dirección MAC e IP (la longitud normal es 6 y 4 Bytes respectivamente)
- Identificador de Mensaje (16 bits): Para indicar si es *REQUEST* o *RESPONSE*
- Direcciones de hardware (MAC) e IP de origen y de destino.

Cada nodo mantiene una memoria denominada ARP *Caché* con las direcciones de los puertos IP y las correspondientes direcciones MAC. Esta es actualizada cada 15 minutos.

Proxy ARP: Un *router* utiliza el *proxy* ARP para ayudar a un *host* en el reconocimiento de direcciones de otras redes y subredes vecinas.

RARP: Funciona con estaciones sin disco duro que no pueden guardar las direcciones IP. Su función es requerir la dirección IP cuando se conoce la dirección MAC. Los protocolos ARP y RARP no utilizan datagramas IP en forma directa, sin embargo generan un datagrama propio de características similares.

Hello: Este protocolo habilita a los elementos de red para reconocer las direcciones MAC mediante paquetes pequeños de presentación. Cuando un nuevo elemento se conecta a la red genera un mensaje *HELLO* en forma *broadcast*, el mismo es emitido en forma periódica para indicar que continúa activo. Para cada mensaje *HELLO* hay un *HELLO REPLY* para configurar la tabla de direcciones.

STP (*Spanning-Tree Protocol*): En las redes construidas mediante *switches Ethernet* se debe cuidar que no ocurran lazos cerrados debido a que los caminos duplicados pueden generar paquetes duplicados. El uso de STP permite eliminar el problema de los lazos cerrados y mantener las ventajas. STP fue desarrollado originalmente en Digital DEC y luego fue incorporado a IEEE 802.1d. Este protocolo permite identificar los lazos cerrados y mantener activo sólo un puerto del *switch*; se asigna a cada puerto un identificador (la dirección MAC y una prioridad).

El STP consiste en un intercambio de mensajes de configuración en forma periódica (entre 1 y 4 segundos) de tal forma que cuando se detecta un cambio en la configuración de la red se recalcula la distancia para asignar un nuevo puerto. Las decisiones se toman en el propio *switch*.

1.6.3.3 PROTOCOLOS DE ENRUTAMIENTO DE CAPA 3

Un área puede contener a todos los *routers* que inician con la misma dirección IP, en este caso el uso de la máscara de red puede ser de utilidad. La expresión dominio se refiere a la porción de red donde los *routers* poseen la misma base de datos topológica. La topología del área es invisible a elementos fuera de ella. El dominio es usado para intercambios con AS.

1.6.3.3.1 PROTOCOLO RIP (*ROUTING INFORMATION PROTOCOL*)

El protocolo RIP es del tipo IGP indicado como inter-AS. La métrica utilizada es del tipo número de saltos (4 bits).

Desde el punto de vista del modelo de capas RIP trabaja sobre UDP con el número de puerto 520; en cambio BGP trabaja sobre TCP (realiza una conexión con control de errores y de flujo). El máximo tamaño del mensaje es de 512 Bytes (sin contar UDP/IP). Los mensajes son:

- *REQUEST*: Pedido de transferencia de la tabla de rutas
- *RESPONSE*: Respuesta al pedido

Normalmente el *REQUEST* es un mensaje con dirección *broadcast*. RIP no puede manejar direcciones con máscara de subred variable.

El paquete del protocolo RIP permite la actualización de las tablas de enrutamiento en los *routers*, el contenido en la dirección IP y el valor de la métrica. La tabla de enrutamiento generada por RIP contiene la siguiente información: Dirección de destino, próximo paso, distancia, *Timer, Flag*.

La tabla de enrutamiento solo mantiene la mejor ruta al destino. Cuando una ruta mejor es detectada se reemplaza la anterior. Cada *router* actualiza un cambio y lo propaga a los demás por lo que el tiempo de convergencia es alto. RIP requiere datos de ruta para actualizar las tablas y periódicamente anuncia la presencia y difunde los cambios que detecta en la red. Utiliza el algoritmo *distance vector*.

Los paquetes de RIP son intercambiados en forma periódica cada 30 segundos (*upgrade)*. Este mecanismo de transmisión periódica carga la red con información de enrutamiento. Si el tiempo de *upgrade* supera el valor de 90 segundos, la ruta de salida se considera inválida. Si se supera el tiempo de 270 segundos (*Flush Timer*), la ruta se elimina de la tabla de enrutamiento.

Es importante notar que el campo de métrica permite un máximo de 15 saltos (*Count Hop*). Si es mayor se considera un destino inalcanzable. Se utiliza un mecanismo denominado *Split Horizont* que permite evitar información sobre rutas que retornan al origen generando lazos de enrutamiento entre 2 nodos.

1.6.3.3.2 PROTOCOLO IGRP (*INTERIOR GATEWAY ROUTING PROTOCOL*)

Este protocolo es original de Cisco para enrutamiento en AS. Es un protocolo que usa el *distance vector* como RIP. Emite la totalidad de la tabla de rutas al inicio y sólo los cambios en períodos preestablecidos como actualización. Mientras RIP usa sólo una métrica con un número de saltos máximo de 15, IGRP utiliza una combinación de métricas: retardo, ancho de banda, confiabilidad y carga del enlace (estos dos últimos se evalúan mediante un número desde 1 a 255).

Todos los *routers* mantienen la tabla de rutas de los vecinos para usarlo en el algoritmo de convergencia. Los tipos de paquetes involucrados se denominan:

- *HELLO*: Paquete *multicast* emitido para indicar la presencia
- *ACKNOWLEDGMENT*: Paquete de reconocimiento
- *UPDATE*: Paquete emitido en forma *unicast* a un nuevo *route*r en la red
- *QUERY, REPLAY* y *REQUEST*: Paquetes que solicitan información en forma *unicast.*

1.6.3.3.3 PROTOCOLO OSPF (*OPEN SHORTEST PATH FIRST*)

Se usa muy frecuentemente como protocolo IGP en redes TCP/IP. Cuando se diseñó se quiso que cumpliera los siguientes requisitos:

- Ser abierto, que no fuera propiedad de una compañía.
- Que permitiera reconocer varias métricas, entre ellas, la distancia física y el retardo.
- Ser dinámico, que se adaptará rápida y automáticamente a los cambio de la topología.
- Ser capaz de realizar el enrutamiento dependiendo del tipo de servicio.
- Ser capaz de equilibrar las cargas dividiendo la misma entre varias líneas.
- Ser capaz de reconocer sistemas jerárquicos pues un único ordenador no puede conocer la estructura completa de Internet.
- Implementar un mínimo de seguridad.

El protocolo OSPF reconoce tres tipos de conexiones y redes:

- Líneas punto a punto entre dos dispositivos de encaminamiento.
- Redes multiacceso con difusión (por ejemplo, la mayoría de redes LAN).
- Redes multiacceso sin difusión (por ejemplo, la mayoría de redes WAN de conmutación de paquetes).

Una red es multiacceso si tiene varios dispositivos de enrutamiento que se pueden comunicar con los demás.

La función del OSPF es encontrar la trayectoria mas corta de un dispositivo de enrutamiento a todos los demás. Cada dispositivo de enrutamiento tiene almacenada en una base de datos la topología de la red de la que forma parte. La representación de esta topología se expresa como un grafo dirigido.

Al arrancar un dispositivo de almacenamiento, este protocolo envía paquetes *HELLO* por todas sus líneas punto a punto y los retransmite a todos los demás dispositivos de enrutamiento. Gracias a las respuestas que recibe sabe cuales

son sus dispositivos de enrutamiento vecinos. El OSPF se basa en el intercambio de información entre los dispositivos de enrutamiento adyacentes, que no es lo mismo que vecinos. Para que no todos los dispositivos tengan que hablar con los demás, se designa uno como adyacente a todos los demás y es éste el que intercambia información con los restantes.

Por motivos de seguridad se determina un dispositivo de enrutamiento como secundario por si el primario cae.

Normalmente, el dispositivo de enrutamiento inunda de mensajes de *LINK STATE UPDATE* a todos sus dispositivos de enrutamiento adyacentes. Estos mensajes tienen un número de secuencia y además para hacerlos confiables son reconocidos por el mensaje *LINK STATE ACKNOWLEDGE*. Además existen otros dos mensajes: *DATABASE DESCRIPTION* que es utilizado para anunciar las actualizaciones que tiene el transmisor, y *LINK STATE REQUEST* que es utilizado para solicitar información a un compañero. Todos los mensajes utilizados en el OSPF se envían como paquetes IP en bruto.

1.6.3.3.4 PROTOCOLO EGP (*EXTERIOR GATEWAY PROTOCOL*)

EGP es el protocolo utilizado para el intercambio de información de enrutamiento entre *gateways* exteriores.

EGP se basa en el sondeo periódico empleando intercambios de mensajes *HELLO/I HEAR YOU*, para monitorear la accesibilidad de los vecinos y para sondear si hay solicitudes de actualización. EGP restringe los *gateways* exteriores al permitirles anunciar sólo las redes de destino accesibles en el AS del *gateway*. De esta forma, un *gateway* exterior que usa EGP pasa información a sus vecinos EGP pero no anuncia la información de accesibilidad de estos fuera del AS. Tiene tres características principales:

- Soporta un protocolo NAP (*Neighbor Acquisition Protocol*). Dos *gateways* se pueden considerar vecinos si están conectados por una red que es transparente para ambos. EGP no especifica la forma en que un *gateway* decide inicialmente que quiere ser vecino de otra. Para convertirse en vecino, debe enviar un mensaje *ACQUISITION CONFIRM* como respuesta a un *ACQUISITION REQUEST*. Este paso es necesario para obtener información de encaminamiento de otro *gateway*.
- Soporta un protocolo NR (*Neighbor Reachability*). El *gateway* lo usa para mantener información en tiempo real sobre la accesibilidad de sus vecinos. El protocolo EGP proporciona dos tipos de mensajes para ese fin: un mensaje *HELLO* y un mensaje *I HEAR YOU*.
- Soporta mensajes de actualización que llevan información de encaminamiento. No se requiere ningún *gateway* para enviar mensajes NR a otro *gateway*, excepto como respuesta a una petición de sondeo (*POLL REQUEST*).

Para realizar estas tres funciones básicas, EGP define 10 tipos de mensajes:

- *ACQUISITION REQUEST:* Solicita que una pasarela se convierta en vecina
- *ACQUISITION CONFIRM:* Respuesta afirmativa a un *ACQUISITION REQUEST*
- *ACQUISITION REFUSE:* Respuesta negativa a un A*CQUISITION REQUEST*

- *CEASE REQUEST:* Solicitud de terminación de la relación de vecindad
- *CEASE CONFIRM:* Confirmación para que cesen las peticiones
- *HELLO:* Solicitud de respuesta de un vecino.
- *I HEAR YOU:* Respuesta el mensaje *HELLO*.
- *POLL REQUEST:* Solicitud de la tabla de encaminamiento de la red
- *ROUTING UPDATE:* Información de accesibilidad de la red
- ERROR: Respuesta a un mensaje incorrecto

1.6.3.3.5 PROTOCOLO BGP (*BORDER GATEWAY PROTOCOL*)

Se encarga de mover paquetes de una red a otra pero en algunos casos debe preocuparse de otras cuestiones que no tienen porque estar relacionadas con el objetivo de mover los paquetes de la forma más eficiente posible.

Los diferentes dispositivos de enrutamiento BGP se comunican entre sí estableciendo conexiones TCP. El protocolo BGP es fundamentalmente un protocolo de *distance vector* en el que cada dispositivo de enrutamiento mantiene el coste a cada destino y la trayectoria seguida. Estos valores son dados periódicamente a cada uno de los vecinos enviando mensajes. La esencia de BGP es el intercambio de información de enrutamiento entre dispositivos de enrutamiento. La información de enrutamiento actualizada se va propagando a través de un conjunto de redes.

BGP involucra tres procedimientos funcionales, que son:

- Adquisición de Vecino: Dos dispositivos de enrutamiento son vecinos si están conectados a la misma subred y se han puesto de acuerdo en que ambos quieren intercambiar regularmente información de enrutamiento. Para llevar a cabo la adquisición de vecino, un dispositivo de enrutamiento envía a otro un mensaje *OPEN*. Si el dispositivo destino acepta la solicitud, devuelve un mensaje *KEEPALIVE* como respuesta.

- Detección de Vecino Alcanzable: Una vez establecida la relación de vecino, para mantener la relación se realiza la detección de vecino alcanzable enviándose periódicamente mensajes *KEEPALIVE*.

- Detección de Red Alcanzable: Para la detección de red alcanzable es necesario que cada dispositivo de enrutamiento tenga una base de datos con todas las redes que puede alcanzar y la mejor ruta para alcanzarla. Cuando se realiza un cambio en la base de datos es necesario enviar un mensaje *UPDATE* por difusión a todos los dispositivos de enrutamiento que implementan BGP para que puedan acumular y mantener la información necesaria.

1.6.3.3.6 PROTOCOLO IS-IS (*INTERMEDIATE SYSTEM - INTERMEDIATE SYSTEM*)

El protocolo de enrutamiento IS-IS está basado en el algoritmo del estado del enlace; además IS-IS permite hacer enrutamiento integrado, es decir calcular las rutas una vez y aplicarlas para todos los protocolos utilizados, permitiendo así auténtico enrutamiento multiprotocolo. Soporta hasta ocho niveles jerárquicos, para reducir así la cantidad de información de enrutamiento intercambiada.

1.6.3.3.7 *TAG SWITCHING*

Es un diseño de Cisco para el *Backbone* de una red IP cuando se trabaja a alta velocidad. Es un avance de la técnica MPLS (*Multiprotocol Layer Switching*).

Luego que las tabla de enrutamiento convergen usando protocolos de enrutamiento convencionales, los distintos *routers,* asignan una etiqueta (*tag*) para cada ruta posible. La *tag* es corta y de longitud fija que es mejor manejada que la tabla de enrutamiento. Las *tags* generadas localmente en el *router* se intercambian con los otros mediante un protocolo TDP (*Tag Distribution Protocol*). Este protocolo permite distribuir, requerir y actualizar la información de *tag*.

El *tag switching* consiste de dos componentes: el reenvío y el control. La información de *tag* se memoriza en una base de datos denominada TIB (*Tag Information Base*). Los paquetes que circulan en la red llevan la *tag* y no requieren de acciones de tabla de enrutamiento. La *tag* puede ser una simple ruta *unicast* o *multicast*, o un identificador de flujo de tráfico. Por otro lado, *tag switching* puede trabajar con QoS mediante información de prioridades.

1.7 ELEMENTOS DE LA CAPA DE TRANSPORTE

Como se ha comentado, existen dos protocolos de transporte en la Internet: TCP es fiable, orientado a conexión con control de flujo, y UDP es no fiable, no orientado a conexión y sin control de flujo. La unidad de transporte de datos de TCP se denomina segmento, y la de UDP mensaje o también datagrama UDP.

TCP prevé una comunicación *full dúplex* punto a punto entre dos *hosts*, no hay soporte para tráfico *multicast*. En UDP la comunicación es *simplex* (aunque obviamente un datagrama UDP puede ser respondido por el receptor con otro); en UDP es posible el tráfico *multicast* o *broadcast*.

1.7.1 PROTOCOLO TCP

Los protocolos de la capa de transporte permiten efectuar las siguientes funciones:

SEGMENTACIÓN y ENSAMBLE

Algunas razones para realizar la segmentación son:

- La distribución del medio de enlace en forma más equitativa entre los usuarios,
- Permite disminuir el retardo de acceso,
- Impide la monopolización del medio de enlace,
- Permite que el tamaño del buffer requerido para los datos sea más pequeño,
- Se logra un control de errores más eficiente cuando los segmentos son pequeños,
- Permite la retransmisión de tramas más cortas.

Sin embargo, puede presentar algunas desventajas:

- Reduce la eficiencia de datos debido a un encabezado proporcionalmente mayor,
- Los bloques pequeños pueden involucrar un número mayor de interrupciones,
- Se requiere un tiempo mayor de procesamiento.

CONTROL DE FLUJO DE DATOS

Referido a la capacidad del receptor de controlar la emisión de datos desde el otro extremo. Los métodos usados son varios, dependiendo de la capa involucrada. Sobre la capa física es un control eléctrico en tanto que sobre las capas superiores se realiza en forma de secuencia de datos.

CONTROL DE ERRORES

Se involucra un campo de bits de paridad (*checksum*) que permite determinar las tramas con error. En este caso se puede efectuar el descarte de la trama de datos afectada o pedir la repetición automática. El *checksum* se calcula como la suma binaria de los datos transmitidos y es menos eficaz que el chequeo de redundancia cíclica CRC.

CONTROL DE CONEXIÓN

El tipo de servicio ofrecido por la red es:

- Orientado a conexión (*Connection-Oriented*),
- No orientado a conexión (*Connection-less Oriented).*

En el primero se establece inicialmente una conexión y se mantiene hasta la desconexión. En el segundo cada mensaje lleva la dirección completa y el enrutamiento es independiente para cada mensaje. Esto puede producir que el arribo de los mensajes esté fuera de orden. En el primero es más fácil el control de flujo para la congestión y el encabezado más corto.

El servicio no orientado a conexión es solicitado por los usuarios de redes de datos debido a que permite el arribo de las tramas aún con interrupción de ciertos enlaces en una red malla. En tanto el servicio orientado a conexión es el generalmente soportado por las empresas de telecomunicaciones por la simplicidad sobre los nodos de enrutamiento. Sin embargo, esta diferencia no es muy apreciable: en orientado a conexión es posible un re-enrutamiento automático y en no orientado a conexión el trayecto en la red es mantenido a menos que exista una razón para cambiarlo.

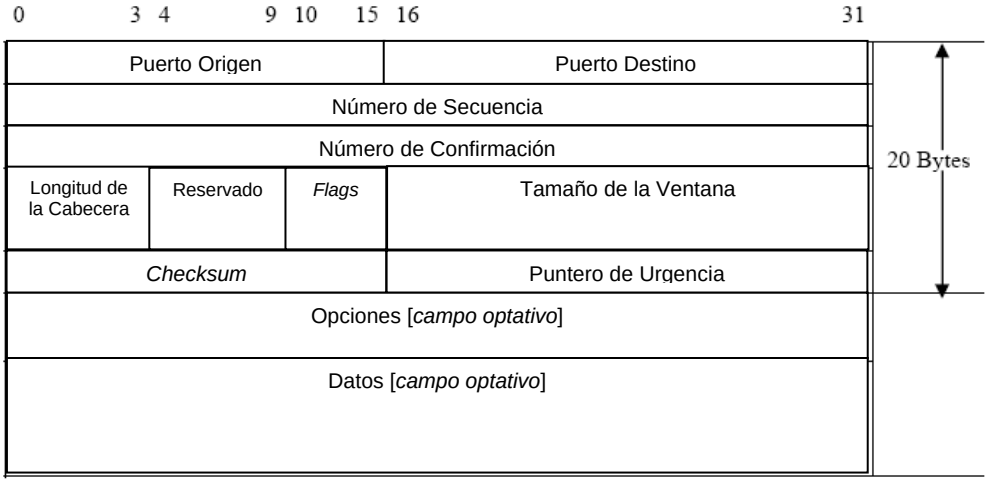
DIRECCIONAMIENTO

En una red de datos se requiere la identificación de los usuarios que participan en la conexión mediante las direcciones de origen y destino. Estas direcciones pueden ser físicas o lógicas.

1.7.1.1 CAMPOS DE TCP

La unidad de datos de este protocolo recibe el nombre de segmento TCP. Como el encabezado debe implementar todos los mecanismos del protocolo su tamaño es bastante grande, como mínimo 20 bytes.

Los campos del encabezado TCP son los siguientes:



- Puerto origen (16 bits): Es el punto de acceso de la aplicación en el origen.
- Puerto destino (16 bits): Es el punto de acceso de la aplicación en el destino.
- Número de Secuencia (32 bits): Identifica el primer *byte* del campo de datos. En este protocolo no se enumeran segmentos sino bytes, por lo que este número indica el primer *byte* de datos que hay en el segmento. Al principio de la conexión se asigna un ISN (*Initial Sequence Number*) y a continuación los bytes son numerados consecutivamente.
- Número de Confirmación (ACK) (32 bits): El protocolo TCP utiliza la técnica de *piggybacking* para reconocer los datos. Cuando el bit ACK está activo, este campo contiene el número de secuencia del primer *byte* que espera recibir. Dicho de otra manera, el número ACK - 1 indica el último bit reconocido.
- Longitud de la Cabecera (4 bits): Indica el número de palabras de 32 bits que hay en la cabecera. De esta manera el TCP puede saber donde se acaba la cabecera y por lo tanto donde empieza los datos. Normalmente el tamaño de la cabecera es de 20 bytes por lo que en este campo se almacenará el número 5. Si el TCP utiliza todos los campos de opciones la cabecera puede tener una longitud máxima de 60 bytes almacenándose en este campo el valor 15.
- Reservado (6 bits): Se ha reservado para su uso futuro y se inicializa con ceros.
- *Flags* (6 bits): Cada uno de los bits recibe el nombre de indicador y cuando está a 1 indica una función específica del protocolo.

    - URG: Hay datos urgentes y en el campo "puntero urgente" se indica el número de datos urgentes que hay en el segmento.
    - ACK: Indica que tiene significado el número que hay almacenado en el campo "número de confirmación".
    - PSH: Sirve para invocar la función de carga (*push*). Como se ha comentado anteriormente con esta función se indica al receptor que debe pasar a la aplicación todos los datos que tenga en la memoria

intermedia sin esperar a que sean completados. De esta manera se consigue que los datos no esperen en la memoria receptora hasta completar un segmento de dimensión máxima. No se debe confundir con el indicador URG que sirve para señalar que la aplicación ha determinado una parte del segmento como urgente.
  - RST: Sirve para hacer un reset de la conexión.
  - SYN: Sirve para sincronizar los números de secuencia.
  - FIN: Sirve para indicar que el emisor no tiene mas datos para enviar.

- Tamaño de la Ventana (16 bits): Indica cuantos bytes tiene la ventana de transmisión del protocolo de control de flujo utilizando el mecanismo de ventanas deslizantes. A diferencia de lo que ocurre en los protocolos del nivel de enlace, en los que la ventana era constante y contaba tramas, en el TCP la ventana es variable y cuenta bytes. Contiene el número de bytes de datos comenzando con el que se indica en el campo de confirmación y que el que envía está dispuesto a aceptar.
- *Checksum* (16 bits): Este campo se utiliza para detectar errores mediante el complemento a uno de la suma en módulo 216 - 1 de todas las palabras de 16 bits que hay en el segmento mas una pseudo-cabecera que se puede ver en la siguiente figura. La pseudo-cabecera incluye los siguientes campos de la cabecera IP: dirección internet origen, dirección internet destino, el protocolo y un campo longitud del segmento. Con la inclusión de esta pseudo-cabecera, TCP se protege a si mismo de una transmisión errónea de IP. Esto es, si IP lleva un segmento a un computador erróneo, aunque el segmento esté libre de errores, el receptor detectará el error de transmisión.
- Puntero de Urgencia (16 bits): Cuando el indicador URG está activo, este campo indica cual es el último *byte* de datos que es urgente. De esta manera el receptor puede saber cuantos datos urgentes llegan. Este campo es utilizado por algunas aplicaciones como TELNET, RLOGIN y FTP.
- Opciones (variable): Si está presente permite añadir una única opción de entre las siguientes:

  - *Timestamp* para marcar en que momento se transmitió el segmento y de esta manera monitorizar los retardos que experimentan los segmentos desde el origen hasta el destino.
  - Aumentar el tamaño de la ventana.
  - Indicar el tamaño máximo del segmento que el origen puede enviar.

1.7.1.2 FUNCIONAMIENTO DE TCP

TCP es un protocolo orientado a conexión. Esto implica que se ha de realizar un paso previo antes de poder intercambiar datos. Este paso es el de establecimiento de conexión.

Este paso es fundamental para poder garantizar la fiabilidad del protocolo, ya que es en este paso previo dónde se obtienen los números de secuencia que permitirán gestionar cualquier intercambio entre lo dos extremos de la comunicación. El método escogido para establecer la conexión se denomina protocolo de 3 pasos (*three way handshake*).

En este esquema, podemos diferenciar un cliente activo que inicia la conexión (*active open*) y un servidor pasivo que tan sólo se limita a contestar (*passive open*) a las peticiones de conexión. Los tres pasos desarrollados en la petición de conexión son:

a. El cliente selecciona un número aleatorio de secuencia. A continuación activa el *flag* de SYN en el campo de opciones. Finalmente envía un segmento TCP al servidor.
b. El servidor recibe la petición y almacena el número de secuencia. Elige un número aleatorio que utilizará como número de secuencia y activa los *flags* SYN y ACK.
   Finalmente envía un segmento con el número de secuencia elegido y con una confirmación del valor recibido más uno.
c. El cliente almacena el número de secuencia. Activa el *flag* de ACK y finalmente envía una confirmación del número recibido más uno.

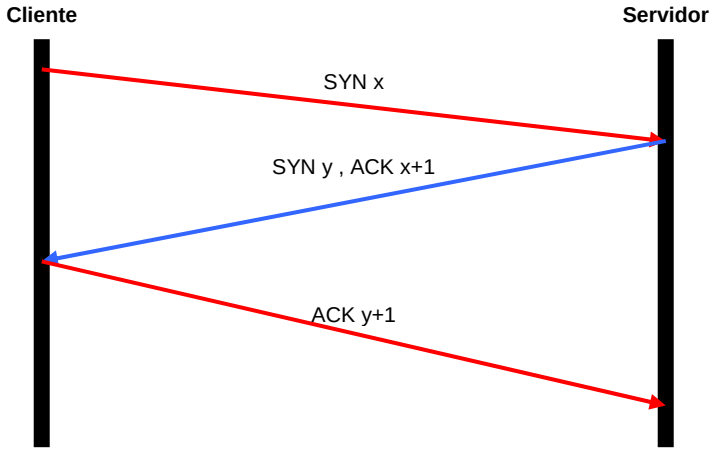


Figura 1.11. Comunicación de Tres Pasos

En el caso de que este tercer paso no se realice, después de un cierto tiempo (entre 70 y 130 segundos dependiendo del sistema operativo) la conexión se liberará.

Esta característica se produce porque cuando se inicia una conexión TCP también se inicializa un temporizador (*timer*), que al finalizar (*time-out*) nos permite liberar la conexión y los recursos asociados.

Debido a la posibilidad de que cualquiera de los dos extremos implicados en la comunicación pueda enviar y/o recibir datos, tenemos la posibilidad de que

cualquiera de los dos extremos finalice la comunicación (enviando una señal FIN) hacia su sentido una vez finalizado el proceso de enviar/recibir datos.

TCP proporciona la capacidad de que cualquiera de los dos extremos de la conexión pueda finalizar su salida (*output*) de datos permitiéndole todavía recibir datos del otro extremo. Esta capacidad se denomina *half-close*.

En un proceso de *half-close* podemos diferenciar de forma clara un extremo activo (*active close*) y un extremo pasivo (*passive close*).

El extremo activo es el que envía la señal de FIN para finalizar ese sentido de la comunicación, mientras que el extremo pasivo se limita a aceptar la petición y enviar un reconocimiento (ACK) de final.

De esta forma tenemos que para finalizar una conexión TCP debemos finalizarla en cada uno de los dos sentidos. Esto obliga a que alternativamente el cliente y el servidor adopten los papeles activo y pasivo en un proceso que consta de 4 fases:

1. (El cliente adopta el papel activo) El cliente decide finalizar la comunicación en su sentido. Enviando al servidor una señal de finalización (FIN) con un número de secuencia.
2. (El servidor adopta el papel pasivo) El servidor recibe esta señal y responde con un reconocimiento (ACK) de señal. Enviando el número de secuencia recibido más uno.
3. Cabe señalar que la finalización (FIN) al igual que la señal de petición de conexión (SYN) consume un número de secuencia. Finalización de la primera *half-close*.
4. (El servidor adopta un papel activo) El servidor decide finalizar la conexión en su sentido y envía una señal de finalización (FIN) de conexión al cliente.
5. (El cliente adopta un papel pasivo) El cliente acepta la petición de finalizar la conexión respondiendo con un ACK y enviando el número de secuencia recibido mas uno. Finalización de la segunda *half-close*. Finalización de la conexión TCP.

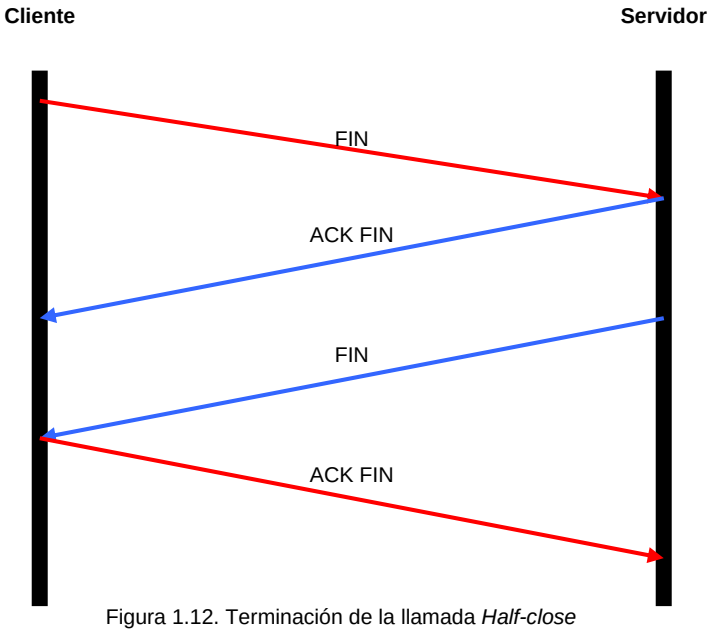


Figura 1.12. Terminación de la llamada *Half-close*

1.7.2 PROTOCOLO UDP

Permite mandar paquetes a través de la red, no es confiable, es decir no garantiza que los paquetes lleguen en el mismo orden que fueron enviados, peor aún no garantiza que los paquetes lleguen a su destino.

Este protocolo es no orientado a conexión, y por lo tanto no proporciona ningún tipo de control de errores ni de flujo, aunque sí utiliza mecanismos de detección de errores. Cuando se detecta un error en un datagrama en lugar de entregarlo a la aplicación se descarta.

Este protocolo se ha definido teniendo en cuenta que el protocolo del nivel inferior (el protocolo IP) también es no orientado a conexión y puede ser interesante tener un protocolo de transporte que explote estas características.

Como el protocolo es no orientado a conexión cada datagrama UDP existe independientemente del resto de datagramas UDP.

El protocolo UDP es muy sencillo y tiene utilidad para las aplicaciones que requieren pocos retardos o para ser utilizado en sistemas sencillos que no pueden implementar el protocolo TCP.

Las características del protocolo UDP son:

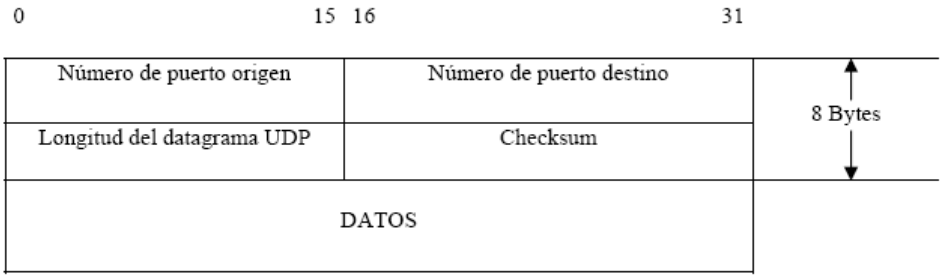
- No garantiza la fiabilidad. No podemos asegurar que cada datagrama UDP transmitido llegue a su destino. Es un protocolo del tipo *best-effort* porque hace lo que puede para transmitir los datagramas hacia la aplicación pero no puede garantizar que la aplicación los reciban.
- No preserva la secuencia de la información que proporciona la aplicación. La información se puede recibir desordenada (como ocurría en IP) y la aplicación debe estar preparada por si se pierden datagramas, llegan con retardo o llegan desordenados.

Como el protocolo UDP es no orientado a conexión y no envía ningún mensaje para confirmar que se han recibido los datagramas, su utilización es adecuada cuando queremos transmitir información en modo *multicast* o *broadcast* pues no tiene sentido esperar la confirmación de todos los destinos para continuar con la transmisión. También es importante tener en cuenta que si en una transmisión de este tipo los destinos enviarán confirmación, fácilmente el emisor se vería colapsado, pues por cada paquete que envía recibiría tantas confirmaciones como destinos hayan recibido el paquete.

Lo que realmente proporciona UDP respecto a IP es la posibilidad de multiplexación de aplicaciones. La dirección del puerto permite identificar aplicaciones gracias a la dirección del puerto.

1.7.2.1 CAMPOS DE UDP

Los campos del encabezado UDP son los siguientes:



- Puerto de Origen (16 bits): Punto de acceso de la aplicación en el origen. Si no se especifica un número de puerto, el campo se fija en 0.
- Puerto de destino (16 bits): Es el punto de acceso de la aplicación en el destino.
- Longitud (16 bits): Indica en bytes la longitud del datagrama UDP incluyendo la cabecera UDP. En realidad es la longitud del datagrama IP menos el tamaño de la cabecera IP. Como la longitud máxima del datagrama IP es de 65.535 bytes y la cabecera estándar de IP es de 20 bytes, la longitud máxima de un datagrama UDP es de 65.515 bytes.
- *Checksum* (16 bits): Suma de complemento de uno del datagrama, incluyendo un pseudoencabezado parecido al del TCP.

1.8 ELEMENTOS DE LA CAPA DE APLICACIÓN

Además de los elementos de red, es muy común la utilización de otros elementos en las redes IP destinados a brindar servicios de seguridad, de comunicaciones, de almacenamiento, etc. y que forman parte de la capa de aplicaciones.

1.8.1 SERVIDOR DE *FIREWALL*

Es un sistema o grupo de sistemas que refuerzan la seguridad en las redes corporativas o proveedores de servicios con protocolos IP. El *firewall* determina los servicios que pueden ser accedidos desde el exterior de la red. Todo el tráfico debe pasar por el *firewall* para ser inspeccionado.

El módulo de *firewall* instalado como un software sobre el *router* o servidor de acceso permite realizar las siguientes funciones:

- **Control de Acceso:** Principal objetivo del *firewall*. Crea un perímetro de defensa diseñado para proteger los recursos. Acepta, rechaza y controla el flujo de paquetes basado en identificadores de capa 3 o aplicaciones. El principio de funcionamiento es que todas las conexiones son denegadas a menos que estén expresamente autorizadas.
- **Logging:** Es el inicio de las conexiones entrantes y salientes. El uso de un sistema *proxy* y *caché* incrementa la velocidad de respuesta de estas operaciones.
- **Traducción de Direcciones:** Permite realizar las funciones de NAT (*Network Address Translator*) asegura la supervisión de la información de entrada y salida.

- **Autentificación:** El proceso de autentificación involucra a 3 componentes: el servidor, el agente y el cliente.
- **Reportes:** El *firewall* ofrece un punto conveniente para monitorear y generar alarmas.

El *firewall* genera dos áreas en una red, el área pública con facilidad de acceso desde el exterior y el área interna, detrás del *firewall* que se encuentra protegida contra la penetración no deseada. El perímetro de defensa se denomina zona desmilitarizada DMZ (*De-Militarized Zone*) y puede ser accedida por un cliente externo. El *firewall* puede trabajar sobre un servidor o sobre un *router*. La ventaja es que se concentra esta acción en un centro de la red consolidado en lugar de estar distribuido en cada *host*. Esta acción es más útil cuando es llevada a cabo por el *router* de entrada a la red.

1.8.2 SERVIDOR DE AUTENTIFICACIÓN

El proceso de autentificar a un cliente puede ser realizado durante 4 etapas posibles: la conectividad del cliente, cuando se accede al software de autentificación del *switch*, mediante un servidor o cuando se autoriza a trabajar en una VLAN (*Virtual LAN*). Existen diversos métodos de autentificación de clientes.

1.8.3 SERVIDOR WEB

En la WWW (*World Wide Web*) se desarrolla el protocolo de hipertexto; por hipertexto se entienden enlaces entre datos, que mediante la selección de palabras y textos resaltados se accede a una otra página adicional. El salto entre páginas es independiente al lugar de almacenamiento. Este proceso se denomina navegación (*Browsing, Cruising o Surfing*) en el ciberespacio.

Se trata de diferentes mecanismos:

- **URL** (*Uniform Resource Locator*): El URL es una forma de identificador de reservas en los servidores *web*. Un URL puede administrar a varios servidores *web* desde un punto al que se dirige el usuario.
- **HTTP** (*Hypertext Transfer Protocol*): Se trata de una arquitectura cliente-servidor, donde el cliente utiliza un visualizador (*Browser Web*). El servidor *web* es uno o más servidores que entregan texto, gráficos, imagen y sonido.
- **HTML** (*Hypertext Markup Language*): Los enlaces de hipertexto se crean mediante el lenguaje HTML. Es una variante de *Standard Generalized Markup Language* (SGML) de ISO-8879
- **CGI** (*Common Gateway Interface*): Esta interfaz define como se comunica el servidor de HTTP con el programa ejecutable mediante un *browser*.
- ***Cookies*:** Se trata de información que el servidor *web* memoriza en el cliente para ser utilizada en una próxima sesión. Puede ser usada para memorizar información de configuración o contraseña de acceso al servidor.

1.8.4 SERVIDOR DE DOMINIOS

La gestión de direcciones IP requiere de una serie de elementos interrelacionados: el servidor DNS permite asociar un nombre de usuario con la dirección IP.

- **DNS** (*Domain Name Service*): Este sistema permite organizar la información de enrutamiento entre un pseudónimo simple de recordar y el número de dirección IP verdadera. El nombre completo tiene como máximo 63 caracteres. De ellos 3 caracteres indican el domino: edu – educación, com – comercial, gov – gubernamental, org – organización, mil – militar; y 2 el país: ar-Argentina, it-Italia, etc. La tabla de dominios memorizada en el servidor se denomina *DNS Caché*.

- **NAT** (*Network Address Translator*): El problema más complejo de Internet es el reducido número de direcciones. La solución a largo plazo es IPv6 con un mayor número de bytes por dirección. Las soluciones instrumentadas sobre IPv4 son dos: el CIDR (*Classless InterDomain Routing*) y el NAT. El proceso NAT propone reducir el número de direcciones IP mediante la reutilización de direcciones privadas en todas las redes. De esta forma una red privada utiliza un direccionamiento propio y el *router* en el borde de la red realiza la función de traducción y direccionamiento hacia la red.

- ***DHCP*** (*Dynamic Host Configuration Protocol*): Cuando un nuevo usuario se agrega a la red o se cambia de posición se requiere asignar una dirección IP y actualizar la base de datos del DNS. El protocolo DHCP fue diseñado para reducir los requerimientos de configuración.

  Además de asignar la dirección IP realiza una configuración automática de los parámetros necesarios para funcionar en la red donde se encuentra. DHCP trabaja sobre TCP y está basado en el protocolo BOOTP, con algunas diferencias. El BOOTP permitía que clientes sin capacidad de memoria puedan funcionar en TCP/IP.

  Se utiliza un modelo Cliente/Servidor por lo que se dispone de uno o varios servidores DHCP. No se requiere de un servidor por subred por lo que el protocolo DHCP debe trabajar a través de *routers*.

  Más de un servidor puede realizar las tareas de asignación de direcciones con el propósito de mejorar la eficiencia del sistema.

- ***DNS UPDATE***: Asociado a DHCP se encuentra el mecanismo *Dynamic DNS Update* que permite la actualización automática del servidor DNS con el nombre y la dirección IP asignada en forma dinámica por el protocolo DHCP. Este protocolo trabaja sobre TCP o UDP mediante peticiones (*request*). El formato del mensaje de actualización (*update*) contiene un encabezado de 96 bits que identifica al que efectúa el requerimiento y diversos campos.

- ***DHCP FAILOVER***: También en sociedad son DHCP se dispone de la técnica *DHCP Failover* que consiste en disponer de servidores duplicados funcionando como pares redundantes. Se dispone de un protocolo de comunicación simplificado para la operación en régimen normal, de interrupción de comunicación entre servidores y de falla del servidor asociado.

1.9 CALIDAD DE SERVICIO (QOS - *QUALITY OF SERVICE*)

Se entiende por calidad de servicio la posibilidad de asegurar una tasa de datos en la red, un retardo y una variación de retardo (*jitter*) acotados a valores contratados con el cliente. Por ejemplo, en las redes Frame Relay o ATM la calidad de servicio se obtiene mediante la Tasa de Información Contratada (*Committed Information Rate*) con el usuario.

Para disponer de una calidad de servicio aceptable en redes soportadas en protocolo IP se han diseñado herramientas a medida como son protocolos de tiempo real y de reservación.

Un problema evidente es que cuando se soporta un servicio de voz sobre IP (VoIP) por ejemplo, los paquetes son cortos y el encabezado es largo comparativamente. En este caso se requiere un encabezado reducido y un proceso de fragmentación e intercalado. Mediante QoS se tiende a preservar los datos con estas características.

Los servicios tradicionales de Internet disponen de una calidad denominada *best-effort,* es decir que la red ofrece el mejor esfuerzo posible para satisfacer los retardos mínimos, lo cual no es mucho pero es suficiente para servicios que no requieren tiempo real. Para servicios del tipo *real time* (voz y vídeo) se requiere una latencia mínima.
Se denomina latencia a la suma de los retardos en la red. Los retardos están constituidos por el retardo de propagación y el de transmisión (dependiente del tamaño del paquete), el retardo por el procesamiento *store and forward* y el retardo de procesamiento (necesario para reconocimiento de encabezado, errores, direcciones, etc).

Un tiempo de latencia variable se define como fluctuación de retardo (*jitter*) sobre los datos de recepción, es decir que los datos llegan al receptor con diferentes retardos, con lo cual los datos de aplicaciones en tiempo real se ven afectados. La solución al *jitter* es almacenar los datos en memorias *buffer*, lo cual introduce un retardo aún mayor. Se han implementado diversas formas de *buffer* garantizados mediante software:

- Cola Prioritaria: donde el administrador de la red define varios niveles (hasta 4) de prioridad de tráfico.
- Cola Definida: donde el administrador reserva un ancho de banda para cada tipo de protocolo específico.
- Cola Ponderada: mediante un algoritmo se identifica cada tipo de tráfico priorizando el de bajo ancho de banda. Esto permite estabilizar la red en los momentos de congestión.

1.9.1 VARIANTES DE SERVICIOS

Los servicios de datos y de multimedia tienen distintos requerimientos de calidad referidos a latencia y *jitter*. Para satisfacer los requerimientos de calidad se acude al manejo de las colas de paquetes, la reservación de ancho de banda y la gestión del tráfico.

Para obtener estos objetivos en diversos ámbitos se han definido variantes de servicios:

- **CoS** (*Class of Service***)**: Se logra mediante 3 bits que se ingresan en un campo adicional de 4 Bytes dentro del protocolo MAC. Estos 3 bits permiten definir prioridades desde 0 (máxima) a 7 (mínima) y ajustar un umbral en el *buffer* de entrada y salida del *switch* para la descarga de paquetes.

- **ToS** (*Type of Service*): Es similar a CoS pero en la capa 3. Sobre el protocolo IP se define el ToS con 3 bits para asignar prioridades. Se denomina señal de precedencia.
- **QoS** (*Quality of Service*): En redes IP se define la Tasa de Acceso Contratada *Committed Access Rate* (CAR) en forma similar al CIR.

1.9.2 CLASIFICACIÓN DE LA QOS

Garantizado: Es utilizado para requerir un retardo máximo extremo-a-extremo. En esta clasificación se reserva un ancho de banda dentro de la red para su uso exclusivo aún en momentos de congestión. Se lo conoce como *Hard QoS*.

Diferenciado: Utiliza la capacidad de dividir el tráfico en la red con múltiples ToS. Se dispone de 3 bits de precedencia para diferenciar las aplicaciones sensibles a la congestión. Es por lo tanto un *Soft QoS*. Su primera línea de defensa frente a la congestión es el uso de *buffer* de datos, lo cual implica el uso de una cola de espera y el retardo correspondiente dependiendo de la prioridad asignada en dicha cola.

***Best-Effort*:** Este es un servicio por omisión que no tiene en cuenta las modificaciones por la QoS. Se trata de una memoria buffer del tipo FIFO (*First In – First Out).*

1.9.3 HERRAMIENTAS PARA PROPORCIONAR QOS

1.9.3.1 MANEJO DE CONGESTIÓN Y TRÁFICO

Las herramientas de que se dispone para asegurar una QoS dentro de una red IP son mecanismos que previenen o manejan una congestión, distribuyen el tráfico o incrementan la eficiencia de la red.

1.9.3.1.1 CONTROL DE CONGESTIÓN EN EL BUFFER DE DATOS

FIFO (*First In – First Out*): El primer mensaje en entrar es el primero en salir. Este es el mecanismo de QoS por omisión en las redes IP. Es válido sólo en redes con mínima congestión. No provee protección, no analiza el ancho de banda ni la posición en la cola de espera.

PQ (*Priority Queuing*): Este mecanismo de control de congestión se basa en la prioridad de tráfico de varios niveles que puede aportar el encabezado del datagrama IP. Se trata de 3 bits disponibles en el segundo *byte* del encabezado de IPv4.

CQ (*Custom Queuing*): Este mecanismo se basa en garantizar el ancho de banda mediante una cola de espera programada. El operador reserva un espacio de *buffer* y una asignación temporal a cada tipo de servicio. Es una reservación estática.

WFQ (*Weighted Fair Queuing*): Este mecanismo asigna una ponderación a cada flujo de forma que determina el orden de tránsito en la cola de paquetes. La ponderación se realiza mediante discriminadores disponibles en TCP/IP y por el ToS en el protocolo IP. En este esquema la menor ponderación es servida primero. Con igual ponderación es transferido con prioridad el servicio

de menor ancho de banda. El protocolo de reservación RSVP utiliza a WFQ para localizar espacios de *buffer* y garantizar el ancho de banda.

1.9.3.1.2 CONTROL DE TRÁFICO

WRED (*Weighted Random Early Detection*): Trabaja monitoreando la carga de tráfico en algunas partes de las redes y descarta paquetes en forma aleatoria si la congestión aumenta. Está diseñada para aplicaciones TCP debido a la posibilidad de retransmisión. Ésta pérdida en la red obliga a TCP a un control de flujo reduciendo la ventana e incrementándola luego en forma paulatina. Un proceso de descarte generalizado, en cambio, produce la retransmisión en olas y reduce la eficiencia de la red. La versión ponderada WRED realiza el desecho de paquetes de forma que no afecta al tráfico de tipo RSVP.

GTS (*Generic Traffic Shaping*): Provee un mecanismo para el control del flujo de tráfico en una interfaz en particular. Trabaja reduciendo el tráfico saliente limitando el ancho de banda de cada tráfico específico y enviándolo a una cola de espera. De esta forma permite un mejor desempeño en topologías con tasa de bits diferentes. Este control de tráfico se relaciona con CAR.

1.9.3.1.3 INCREMENTO DE LA EFICIENCIA

LFI (*Link Fragmentation and Interleaving*): El tráfico interactivo como Telnet y VoIP es susceptible de sufrir latencia y *jitter* con grandes paquetes en la red o largas colas en enlaces de baja velocidad. Se basa en la fragmentación de datagramas y el intercalado de los paquetes de tráfico.

RSVP (*Resource Reservation Protocol*): Se trata de implementar el concepto de señalización. Se dispone de dos tipos de señalización: en banda, por ejemplo los bits de precedencia para ToS y fuera de banda, mediante un protocolo de comunicación como el RSVP. Este protocolo permite que un *host* o *router* asegure la reservación de ancho de banda a lo largo de la red IP.

RTP-HC (*Real-Time Protocol-Header Compression*): La compresión del encabezado permite mejorar la eficiencia del enlace en paquetes de corta carga útil. Se trata de reducir los 40 bytes de RTP/UDP/IP a una fracción de 2 a 5 bytes, eliminando aquellos que se repiten en todos los datagramas.

No todas las herramientas disponibles son usadas en los mismos *routers*. Por ejemplo, la clasificación de paquetes, el control de admisión y el manejo de la configuración se usan en los *routers* de borde, en tanto que en el *backbone* se gestiona la congestión. El tratamiento de la congestión se fundamenta en el manejo de las colas en *buffer* mediante diferentes técnicas. El *buffer* es la primera línea de defensa frente a la congestión. El manejo correcto del mismo permite determinar el servicio de calidad diferenciada. Una segunda defensa es el control de flujo. El problema del control de flujo en TCP es que se ha planea de extremo a extremo y no considera pasos intermedios.

En TCP cada paquete de reconocimiento (*Acknowledgment*) lleva un crédito con el tamaño del *buffer* disponible por el receptor. Un sobreflujo de datos en los *routers* de la red se reporta mediante el mensaje *Source Quench* en el protocolo ICMP. Estos mecanismos son ineficientes y causan severos retardos en la conexión.

1.9.3.2 PRIORIZACIÓN DE TRÁFICO

ToS

Los estándares IEEE 802.10 y 802.1Q fueron propuestos para el manejo de las redes VLAN, este último es el utilizado con regularidad. En el estándar 802.1Q se define el VLAN *Tagging Switch* que permite una identificación de la VLAN y la posibilidad priorizar el servicio. La trama del paquete en capa MAC incluye 32 bits adicionales al IEEE 802.3 que se colocan luego de las direcciones MAC y antes del *Type/Length*.

Los 32 bits son:

- **TPID** (*Tag Protocol Identifier*): 16 bits usados para identificar el protocolo.
- **TCI** (*Tag Control Information*): 16 bits.
  - **UP** (*User Priority*): 3 bits. Se trata de CoS desde 0 a 7. Esta información permite poner en práctica la CoS definida en IEEE 802.1p.
  - **CFI** (*Canonical Format Indicator*): 1 bit para ser usado por Token Ring.
  - **VLANI** (*VLAN Identifier*): 12 bits que permiten identificar cada VLAN (1-1005). Permite la operación entre diferentes redes.

QoS

El campo de prioridad en el encabezado de IPv4 permite definir varios ToS. Se trata de 3 bits que por razones históricas tienen diferentes nombres (*routing*, *priority*, etc) y que pueden ser usados para asignar prioridad. Se aplica un control de acceso extendido (EACL – *Extended Access Control*) para definir la política de la red en términos de congestión. En redes heterogéneas se deben convertir estos ToS en equivalentes.

CAR

Se ofrece especificando políticas de tráfico y ancho de banda. El umbral de CAR se aplica al puerto de acceso para cada puerto IP o por flujo de aplicación individual. Algunas opciones de CAR son:

- Política de prioridad:
  - Máximo: El exceso de ancho de banda es descartado
  - Premium: El exceso es señalado con un nivel de preferencia más bajo
  - Mejor Esfuerzo: Por encima de un umbral se cambia la preferencia y sobre otro los paquetes son eliminados

- Política de asignación:
  - Por Aplicación: Diferentes políticas son usadas en distintas aplicaciones.
  - Por Puerto: Los paquetes que ingresan por un puerto son clasificados con alto nivel de prioridad.
  - Por Dirección: Puede diferenciarse entre la dirección IP de origen y destino y asignar la prioridad en cada caso.

FORMACIÓN DE ANILLOS

Se trata de configurar la red de *switches* con enlaces en lazos para incrementar la redundancia. Los lazos están prohibidos en Ethernet pero mediante el protocolo STP se pueden identificar los lazos y mantener activo sólo un puerto del *switch*. Por otro lado, utiliza un algoritmo que permite identificar el mejor camino libre de circuitos en la red de *switches*.

Para lograr este objetivo, se asigna a cada puerto un identificador consistente en la dirección MAC y una prioridad. La selección del puerto se puede asignar en términos de prioridad (0-63) y costo (0-65535).

El STP consiste en un intercambio de mensajes de configuración en forma periódica (entre 1 y 4 seg). Cuando se detecta un cambio en la configuración de la red (por falla o cambio de costo del puerto) se recalcula la distancia para asignar un nuevo puerto. Las decisiones se toman en el propio *switch*. En condiciones normales se selecciona un *switch* para que trabaje como *Switch* Raíz para determinar una topología de red estable. Por omisión el *switch* que posee la dirección MAC más baja es el seleccionado como Raíz.

1.9.4 PROTOCOLOS PARA ASEGURAR LA QOS PARA APLICACIONES DE TIEMPO REAL

RSVP (*Resource Reservation Protocol*). Este protocolo permite que un *host* o un *router* asegure la reservación de ancho de banda a lo largo de la red IP. Es del tipo orientado al receptor (el receptor solicita la reservación) y es útil para aplicaciones de tipo *símplex*. Puede funcionar como *unicast* o *multicast*.

RTP (*Real-Time Transport Protocol*). Se utiliza sobre el protocolo UDP para aplicaciones como H.323 o VoIP.

RTCP (*Real-Time Transport Control Protocol*). Este protocolo se utiliza para control de calidad de servicio sobre aplicaciones que trabajan sobre RTP.

IGMP (*Internet Group Management Protocol*). Este protocolo se utiliza para aplicaciones del tipo *multicast* cuando se requiere distribuir la misma información sobre un grupo de usuarios y reduciendo el ancho de banda ocupado. Se emite un mismo paquete con dirección *multicast* en lugar de uno para cada dirección *unicast*.

1.9.4.1 PROTOCOLO PARA LA RESERVACIÓN DE ANCHO DE BANDA (RSVP)

Los servicios del tipo SMTP o FTP en Internet son con calidad mejor esfuerzo, es decir, no prevén una calidad de servicio. Esto tiene como consecuencia una latencia variable y jitter sobre la información del tipo tiempo-real (audio o vídeo).

RSVP permite la reservación de ancho de banda para asegurar una QoS. El protocolo RSVP trabaja en conjunto con el protocolo de transporte RTP para servicios de voz y vídeo en tiempo-real.

Existen dos formas de reservación del ancho de banda:

- **Estática:** Se asigna un porcentaje fijo del canal de comunicación a cada tipo de protocolo (por ejemplo, 10% a HTTP, 15% a FTP, 3% a Telnet, etc).
- **Dinámica:** El protocolo RSVP permite reservar el ancho de banda en esta forma para asegurar una calidad de servicio QoS en las redes IP.

El protocolo RSVP se define para los servicios integrados en Internet. Es utilizado por el *host* para solicitar una QoS al *router* para una aplicación particular y es usado por el *router* para establecer un ancho de banda con todos los nodos intermedios del trayecto.

Opera tanto sobre IPv4 como sobre IPv6; no es un protocolo de enrutamiento y sólo se utiliza para reservar ancho de banda y *buffer*. En el modelo de capas el protocolo RSVP ocupa la función de la capa 3 sobre IP, en la misma forma que los protocolos de enrutamiento (OSPF y BGP), de *multicast* (IGMP), de gestión (ICMP) y de transporte (TCP y UDP).

Una sesión manejada por del protocolo RSVP está definida mediante 3 direcciones: la dirección IP de destino, el identificador de protocolo y el puerto UDP. En el caso de operar con protocolo *multicast* primero se establece el enlace mediante IGMP y luego mediante RSVP.

1.9.4.1.1 CONTROL DE TRÁFICO

La QoS es implementada por un mecanismo de flujo de datos denominado control de tráfico. Este mecanismo incluye 3 etapas:

- Clasificación: Para determinar la QoS y la ruta de cada paquete,
- Control de Admisión: Para asegurar la disponibilidad de la reservación y
- Proceso de Determinación Temporal de Emisión (*Packet Scheduler*).

El requerimiento de reservación es iniciado por *host* receptor y pasa por los distintos *routers* de la red. Si algún mecanismo intermedio falla se genera un reporte de error.

Este protocolo mantiene mediante *software* el estado de los *routers* y *host*, entregando un soporte dinámico para cambios de miembros y adaptación automática de cambios de enrutamiento. No es un protocolo de enrutamiento pero depende de los mismos.

CAPÍTULO 2
VOIP

2.1 INTRODUCCIÓN

Hace 30 años no existía Internet, y las comunicaciones se realizaban por teléfono a través de la Red Telefónica Pública Conmutada (PSTN – *Public Switched Telephone Network*), pero con el pasar de los años y el avance tecnológico han ido apareciendo nuevas tecnologías y aparatos bastante útiles que nos han permitido pensar en nuevas tecnologías de comunicación: PCS (*Personal Communication Service*), teléfonos celulares y finalmente la popularización de la gran red Internet.

Hoy por hoy podemos ver una gran revolución en comunicaciones, todas las personas usan las computadoras e Internet en el trabajo y en el tiempo libre para comunicarse con otras personas, para intercambiar datos y a veces para hablar con mas personas usando aplicaciones como NetMeeting o teléfono IP (*Internet Phone*), el cual particularmente comenzó a difundir en el mundo la idea que en el futuro se podría utilizar una comunicación en tiempo real por medio de la PC: VoIP (*Voice over Internet Protocol*).

Después de haber constatado que desde una PC con elementos multimedia, es posible realizar llamadas telefónicas a través de Internet, se podría pensar que la telefonía IP es algo más que un juguete, pues la calidad de voz que se obtiene a través de Internet es muy pobre.

Este gran avance de transportar la voz a través de Internet ha llevado al desarrollo de normas y protocolos que permiten garantizar el tráfico de voz sobre este tipo de redes, y han llevado a la disponibilidad de productos comerciales de alta calidad que permiten la implementación de sistemas telefónicos basados en IP con una calidad similar a la proporcionada por la PSTN.

No obstante, si en una empresa se dispone de una red de datos que tenga un ancho de banda bastante grande, también se podría pensar en la utilización de esta red para el tráfico de voz entre las distintas sucursales de la empresa. Las ventajas que se obtendrían al utilizar la red para transmitir tanto la voz como los datos son evidentes, ahorro de costos de comunicaciones, pues las llamadas entre las diversas sucursales de la empresa saldrían gratis.

El crecimiento y la fuerte implantación de las redes IP, tanto en el ámbito local como en el global, el desarrollo de técnicas avanzadas de digitalización de voz, mecanismos de control y priorización del tráfico, protocolos de transmisión en tiempo real, así como la implementación comercial de estándares que permiten QoS en redes IP, han creado un entorno que hacen posible el transportar la voz sobre IP.

Aunque VoIP existe desde hace algunos años, la demanda de servicios ha forzado una rápida evolución de la tecnología. El uso de servicios de banda ancha y la integración de voz y datos a todos los niveles, han favorecido el desarrollo de aplicaciones y protocolos VoIP, que han sido pensados para ofrecer un mayor conjunto de características, escalabilidad y estandarización mejorados para brindar el servicio de telefonía.

La mayoría de las redes existentes de proveedores de servicios (ISP – *Internet Service Provider*) soportan principalmente servicios de datos basados en IP. Estos ISPs poseen y siguen desplegando infraestructuras IP que les permitirán incorporarse al mercado de los servicios de voz sobre IP. Los mayores proveedores de telecomunicaciones están buscando la forma de recortar el costo de funcionamiento y el mejoramiento de las redes de voz existentes, y lograr reemplazar y aumentar sus redes con soluciones de VoIP. Lo mismo está ocurriendo en las redes corporativas que están tratando de aprovechar mejor sus recursos.

Además de las ventajas del costo, los servicios de VoIP tienen ventajas técnicas importantes sobre la telefonía de conmutación de circuitos. Las redes VoIP se basan en una arquitectura abierta, significando esto que los servicios de VoIP son más intercambiables y más modulares que los de un sistema de conmutación de circuitos. Es posible seleccionar productos sin estar atado a un proveedor específico. Los estándares abiertos permiten también que la realización de nuevos servicios puedan desarrollarse y desplegarse rápidamente, en lugar de esperar que un productor particular desarrolle una solución propietaria.

Un sistema de VoIP tiene 3 funciones básicas que son: digitalización de la voz, empaquetamiento de la voz, y enrutamiento de paquetes. Si la finalidad es conectarse a la PSTN, se requiere adicionalmente un mecanismo para convertir las direcciones IP a números telefónicos y realizar la operación inversa, utilizando el método de señalización adecuado a la red telefónica con la cual se está conectando.

Existen estándares que cubren cada uno de éstos aspectos, algunos provenientes de la telefonía, como los CODECs (COdificadores – DECodificadores) utilizados para digitalizar la voz, y otros provenientes de la transmisión de datos, como los protocolos de transmisión de paquetes. Además hay estándares que permiten la interconexión entre teléfonos y PCs en redes IP los cuales satisfacen las funciones básicas de un sistema de telefonía de voz sobre IP.

Uno de los estándares utilizado en VoIP es el H.323 desarrollado por la ITU, éste no se puede considerar como un protocolo en sí mismo, más bien se trata de una especificación que define el modo de interactuar de varios protocolos entre sí.

Una red H.323 requiere intercambiar una gran cantidad de mensajes para establecer una simple llamada, y su complejidad crece cuando se ven involucrados varios segmentos de red.

Con el objeto de crear un protocolo mucho más simple, el grupo de trabajo del IETF (*Internet Engineering Task Force*) dio lugar al protocolo SIP (*Session Initiation Protocol*), el cual requiere menos código en su implementación que H.323 lo que reduce los requerimientos de memoria de los equipos involucrados.

SIP es un estándar destinado al establecimiento, modificación y liberación de sesiones multimedia, orientado a la creación de servicios de voz sobre IP. Al tratarse de un estándar proveniente del mundo de Internet, se convierte en una propuesta cercana y adecuada a las necesidades de telefonía de VoIP. SIP utiliza un modelo transaccional similar al de HTTP, intercambio de mensajes codificados en ASCII, y una arquitectura basada en un modelo punto a punto, lo cual lo hace un protocolo simple al utilizar un menor número de mensajes al momento de establecer una llamada, escalable y altamente extensible debido a que se encuentra complementado para la incorporación de otros servicios y tecnologías conforme vayan apareciendo.

2.1.1 ESCENARIOS DE IMPLEMENTACIÓN

Deben distinguirse dos escenarios de aplicación de la voz IP en servicios de telefonía. El primero es cuando la voz IP es transportada a través de redes privadas empresariales y el segundo, cuando la red de transporte usada entre los dos extremos de la conversación es Internet. En el primer el caso, ésta se considera voz viajando sobre el protocolo IP, mas no en Internet, este último caso es cuando hablamos de telefonía por Internet. La diferencia entre los dos escenarios no son únicamente el medio de transporte sino también las posibilidades de establecer mecanismos de control de calidad que garanticen la misma calidad en todo momento.

Los mecanismos y las técnicas aplicadas en ambos casos son diferentes pero los niveles de calidad que se consiguen son muy similares y, en algunos casos, superiores a la telefonía convencional. Por ejemplo, los proveedores de servicio que utilizan Internet como una red de transporte para voz IP usan una técnica a partir de software que evita que los puntos de congestión causen pérdidas de calidad. Cuando se trata de transportar la voz IP a través de redes privadas hay medios más simples y efectivos que aseguran que los paquetes de voz se dirigen a cada uno de los dispositivos de la red antes que los datos y así, se evitan potenciales atrasos en caso de saturación de la red.

LLAMADAS TELÉFONO A TELÉFONO

En este caso tanto el origen como el destino necesitan ponerse en contacto con un *Gateway*. Suponiendo que el teléfono A descuelga y solicita efectuar una llamada al teléfono B. El *Gateway* de A solicita información al *Gatekeeper* sobre como alcanzar a B, y éste le responde con la dirección IP del *Gateway* que da servicio a B. Entonces el *Gateway* de A convierte la señal analógica del teléfono A en un caudal de paquetes IP que encamina hacia el *Gateway* de B, el cuál va regenerando la señal analógica a partir del caudal de paquetes IP que recibe con destino al teléfono B. Es importante notar que el *Gateway* de B se encarga de enviar la señal analógica al teléfono B.

Por tanto, se tiene una comunicación telefónica convencional entre el teléfono A y el *Gateway* que le da servicio (*Gateway* A), una comunicación de datos a través de una red IP, entre el *Gateway* A y el *Gateway* B, y una comunicación telefónica convencional entre el *Gateway* que da servicio al teléfono B (*Gateway* B), y éste. Es decir, dos llamadas telefónicas convencionales, y una comunicación IP. Si las dos primeras son llamadas locales, el margen con respecto a una llamada telefónica convencional de larga distancia nacional o internacional, es muy grande.

LLAMADAS PC A TELÉFONO O VICEVERSA

En este caso sólo un extremo necesita ponerse en contacto con un *Gateway*. La PC debe contar con una aplicación que sea capaz de establecer y mantener una llamada telefónica. Suponiendo que una PC A trata de llamar a un teléfono B. En primer lugar la aplicación telefónica de A solicita la información al *Gatekeeper*, que le proporcionará la dirección IP del *Gateway* que da servicio a B. Entonces la aplicación telefónica de A establece una conexión de datos, a través de la red IP, con el *Gateway* de B, el cuál va regenerando la señal analógica a partir del caudal de paquetes IP que recibe con destino al teléfono B. De nueva cuenta, el *Gateway* de B es el encargado de enviar la señal analógica al teléfono B.

Por tanto, se tiene una comunicación de datos a través de una red IP, entre la PC A y el *Gateway* de B, y una comunicación telefónica convencional entre el *Gateway*

que da servicio al teléfono B (*Gateway* B), y éste. Es decir, una llamada telefónica convencional, y una comunicación IP.

LLAMADAS PC A PC

Este caso es diferente. Ambas PCs sólo necesitan tener instalada la misma aplicación encargada de gestionar la llamada telefónica, y estar conectados a la red IP, para poder efectuar una llamada IP. Al fin y al cabo es como cualquier otra aplicación Internet.

2.1.2 VENTAJAS DE VOIP

El empleo de VoIP garantiza el obtener grandes ahorros económicos puesto que se integran dos infraestructuras de comunicación, una de datos y otra de voz, en una sola, presentando esta última mayor escalabilidad así como un mantenimiento más sencillo, pero además, genera la incorporación de nuevos servicios que incluyen a la vez voz y datos, y que proporcionan al usuario final mejores herramientas de comunicación.

Las ventajas principales pueden resumirse en las siguientes categorías:

- **REDUCCIÓN DE COSTOS**: Al compartir equipos y costos de operación entre las redes de voz y de datos, es evidente que se puede mejorar la eficacia de la red, puesto que el exceso de ancho de banda en alguna de las redes, puede ser utilizado por la otra. Además de que en algunos casos se pueden evitar costos de conexión a la PSTN, incluyendo también el caso de llamadas de larga distancia entre sucursales estatales.

- **SIMPLIFICACIÓN**: VoIP permite que se integren aplicaciones telefónicas, de fax, correo electrónico y otras aplicaciones para capitalizar los beneficios de una mensajería unificada. Este tipo de sistemas podría evitar interrupciones innecesarias, mientras se asegura que el usuario final reciba la información en la forma más conveniente, sea cual fuere su ubicación alrededor del mundo.

- **CONSOLIDACIÓN:** La amplia difusión de los protocolos empleados por IP, para diferentes servicios ya existentes, permite una complejidad reducida y una mayor flexibilidad para la implementación de nuevos servicios de comunicaciones. Las redes de tipo IP son las más empleadas ya sea para Internet, Intranets o Extranet, permitiendo así que VoIP se integre en las redes existentes como un servicio más. Además, cuenta con estándares que permiten la interoperabilidad de equipos de diferentes fabricantes.

- **USOS AVANZADOS:** Aunque las principales aplicaciones de VoIP se basan en la telefonía básica y el fax, puede emplearse para implementar prácticas de funcionamiento más flexibles, en las cuales los miembros de un equipo de trabajo pueden participar desde el hogar o desde cualquier parte del mundo, creando así "equipos virtuales". Empleando la red VoIP, los miembros pueden ver cuando sus colegas ingresan a la red LAN, o utilizan el teléfono. VoIP ofrece capacidades mejoradas para el empleo del ancho de banda, y hace que la video conferencia sea una opción viable y redituable para discusiones entre los miembros dispersos del equipo de trabajo.

2.1.3 APLICACIONES DE VOIP

Existen una gran variedad de aplicaciones desarrolladas exclusivamente para VoIP, que generan grandes ventajas en el ámbito laboral y genera un ahorro de costos significativo.

MULTICONFERENCIA

La telefonía IP permite la conexión de varios usuarios simultáneamente, compartiendo conversaciones de voz e incluso documentos sobre el que todos los miembros de la conferencia pueden participar. Esta aplicación resulta de gran utilidad, puesto que se tiene la posibilidad de generar equipos de trabajo virtuales.

ACCESO REMOTO

Los usuarios pueden acceder a los servicios corporativos de comunicaciones, ya sean voz, datos o fax, empleando la Intranet de su compañía a través de los teléfonos IP. Por ejemplo, se podría emular una extensión remota para un PBX, mediante VoIP.

MOVILIDAD

Las llamadas pueden ser realizadas y recibidas en cualquier PC multimedia conectada a Internet, de igual forma que se haría en el lugar del trabajo del usuario, independientemente de su ubicación geográfica.

COMUNICACIONES UNIFICADAS

La integración del correo electrónico, fax, correo de voz en un sistema de mensajería mejorado, permitiendo al usuario recibir dicho mensaje en la forma más conveniente para él.

ASISTENTE PERSONAL VIRTUAL

Un servicio de voz en tiempo real que puede reenviar llamadas basadas en IP, así como correos electrónicos al usuario en cualquier parte de la red que se encuentre. Así como almacenar llamadas para ser escuchadas posteriormente por el destinatario.

REDES PRIVADAS VIRTUALES DE VOZ

Consistente en la interconexión de los PBX a través de la red IP corporativa, de tal suerte que se puede realizar una llamada desde una extensión en el departamento A hasta otra extensión en el departamento B empleando la red de datos de la empresa, produciéndose esta llamada de forma gratuita.

Las aplicaciones nombradas hasta el momento, están enfocadas en mejorar las habilidades de los usuarios para administrar sus comunicaciones, ya sean en forma de voz, correo electrónico, fax, etc. así como mejorar la colaboración entre usuarios. Otras aplicaciones están diseñadas para facilitar una interacción flexible entre una compañía y sus consumidores, usando por ejemplo:

CENTROS DE LLAMADAS VÍA WEB (COMERCIO ELECTRÓNICO HABILITADO POR VOZ)

Los usuarios que visiten una página *web* con esta aplicación, podrían no sólo visualizar la información ofrecida, sino que podrían establecer una comunicación de voz en tiempo real con un representante de ventas sin la necesidad de cortar la conexión, de tal forma que el operador de ventas, al momento de establecer la llamada, tendrá en su pantalla la misma información visualizada por el usuario. De esta forma, se consiguen sistemas de gran calidad en el servicio, además de que reduce el costo de líneas telefónicas y de Distribuidores Automáticos de Llamadas

CONTENIDO WEB ACTIVADO POR VOZ

Los usuarios pueden seleccionar contenido *web* por medio de teléfonos tradicionales o inalámbricos. Empleando sencillas palabras habladas, los usuarios de este servicio, pueden navegar a través de menús verbales, y así obtener información del tráfico, noticias, deportes, etc, sin tener que acceder por medio del teclado del teléfono.

Además, existen características que simplemente mejoran la funcionalidad de los servicios telefónicos y pueden proveer:

SEGUNDA LÍNEA VIRTUAL

Se tiene la posibilidad de realizar llamadas IP desde la PC, así como recibir correos de voz, faxes, notificación de llamada en espera e identificación de llamadas, todo mediante una sola línea telefónica.

DISCADO ACTIVADO POR VOZ

Agenda independiente basada en red, que puede ser activada por el usuario desde cualquier parte del mundo, mediante cualquier dispositivo.

LLAMADA DE INTERNET EN ESPERA

Es un servicio que notifica a los usuarios cuando se le está tratando de localizar. Cuenta con identificador de llamada y la opción de elegir el modo de responder.

2.2 ARQUITECTURA Y FUNCIONAMIENTO

La tecnología VoIP permite establecer comunicaciones de voz empleando como medio de transporte una red IP. Es así que, en lugar de utilizar la infraestructura telefónica tradicional de conmutación de circuitos, la voz es transmitida en forma de paquetes por medio de una red IP.

Años atrás se descubrió que mandar una señal a un destino remoto también podía hacerse también de manera digital: antes de enviar la señal se debía digitalizar con un ADC (*Analog to Digital Converter*), transmitirla y en el extremo de destino transformarla de nuevo a formato análogo con un DAC (*Digital to Analog Converter*).

VoIP funciona de esa manera, digitalizando la voz en paquetes de datos, enviándola a través de la red y reconvirtiéndola a voz en el destino. Básicamente el proceso comienza con la señal análoga del teléfono que es digitalizada en señales PCM (*Pulse Code Modulation*) por medio del CODEC.

Las muestras PCM son pasadas al algoritmo de compresión, el cual comprime la voz y la fracciona en paquetes que pueden ser transmitidos, para este caso a través de una red privada WAN. En el otro extremo de la nube se realizan exactamente las mismas funciones en un orden inverso.

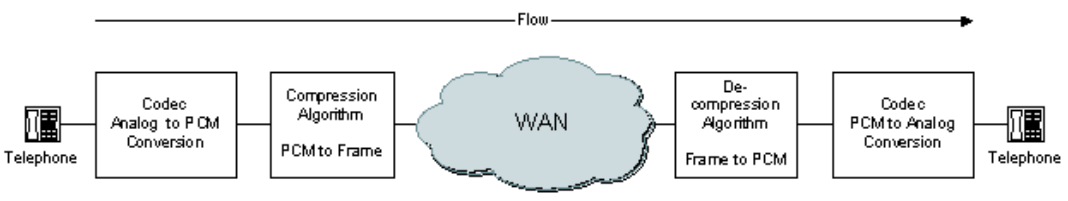


Figura 2.1. Comunicación a través de una WAN

Dependiendo de la forma en la que la red este configurada, el *Router* puede realizar la labor de codificación, decodificación y/o compresión. Por ejemplo, si el sistema usado es un sistema análogo de voz, entonces el *Router* realiza todas las funciones mencionadas anteriormente de la siguiente manera

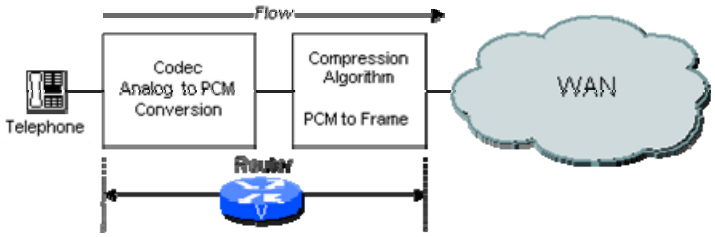


Figura 2.2. Comunicación a través de Router

Si, por otro lado, el dispositivo utilizado es un PBX (*Private Box eXchange*) digital, es entonces éste el que realiza la función de codificación y decodificación, y el *Router* sólo se dedica a procesar las muestras PCM que le ha enviado el PBX.

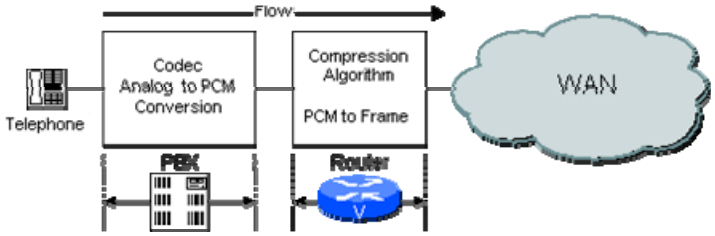


Figura 2.3. Comunicación a través de PBX

Para el caso en el que el transporte de voz se realiza sobre la red pública Internet, se necesita una interfaz entre la PSTN y la red IP, el cual se denomina *Gateway* y es el encargado en el lado del emisor de convertir la señal analógica de voz en paquetes IP para ser transportados a través de la red, del lado del receptor su labor es inversa, dado que descomprime los paquetes IP que recibe de la red de datos, y recompone el mensaje a su forma analógica original conduciéndolo de nuevo a la PSTN en el sector de la última milla para ser transportado al destinatario final y ser reproducido por el parlante del receptor.

Todas las redes deben tener de alguna forma las características de direccionamiento, enrutamiento y señalización. El direccionamiento es requerido para identificar el origen y destino de las llamadas, también es usado para asociar clases de servicio a cada una de las llamadas dependiendo de la prioridad. El enrutamiento por su parte encuentra el mejor camino a seguir por el paquete desde la fuente hasta el destino y transporta la información a través de la red de la manera más eficiente, la cual ha sido determinada por el diseñador. La señalización alerta las estaciones terminales y a los elementos de la red su estado y la responsabilidad inmediata que tienen al establecer una conexión.

Inicialmente, la voz sobre redes IP, fue implementada para reducir el ancho de banda mediante la compresión vocal y en consecuencia, disminuir los costos en el transporte internacional. Sin embargo, rápidamente migró a una red de servicios integrados sobre la misma LAN. Posteriormente, esta tecnología migró de la LAN a la WAN, con el nombre de Telefonía sobre IP (ToIP – *Telephony Over Internet Protocol*). Los operadores de este tipo de servicio son denominados ITSP (*IP Telephony Service Provider*) por similitud a los ISP de Internet.

Una diferencia importante entre VoIP y ToIP es la interoperatividad con las redes telefónicas actuales y los servicios de valor agregado que generalmente se brindan en las PSTN soportadas en señalización SS7 y redes inteligentes.

2.2.1 FUNCIONES BÁSICAS DE VOIP

Los componentes de VoIP deben ser capaces de realizar las mismas funciones que la PSTN.

SEÑALIZACIÓN

La señalización en las redes VoIP es tan crítica, como lo es en la PSTN. Dicha señalización activa y coordina los diferentes componentes para completar una llamada. Aunque la naturaleza de la señalización es la misma, existen algunas diferencias técnicas y arquitectónicas en la red VoIP.

La señalización en una red VoIP es completada mediante el intercambio de datagramas IP entre los componentes. El formato de estos mensajes es determinado por un conjunto de protocolos estándares. Sin importar los protocolos empleados, este flujo de mensajes es indispensable para el funcionamiento de una red de voz y pueden necesitar un tratamiento especial para garantizar su entrega.

SERVICIOS DE BASES DE DATOS

Estos servicios son empleados para localizar un punto final y traducir la dirección que emplean dos redes heterogéneas. Por ejemplo, la PSTN emplea números telefónicos para identificar los puntos finales, mientras que la red VoIP emplea direcciones IP y números de puertos a modo de identificar un punto final. Una base de datos de control de llamada contiene estos puertos y traducciones, además de generar reportes de transacciones para propósitos de tarificación. También es capaz de emplear lógica adicional para proporcional seguridad en la red, como puede ser, negar a un punto terminal específico, realizar llamadas internacionales en el extremo de la PSTN. Esta funcionalidad, adicionada con el control de estado de las llamadas, coordina la actividad de los elementos en una red VoIP.

CONTROL DE CARGA

La conexión de una llamada es realizada por dos puntos finales que abren sesiones de comunicación entre cada uno. En la PSTN, el conmutador público o el PBX, conectan los canales lógicos a través de la red para completar la llamada. En una implementación VoIP, dicha conexión es un flujo multimedia (audio, video o ambos) transportado en tiempo real. Esta conexión es el canal del portador y representa el contenido de voz o video a ser entregado. Cuando la comunicación se completa, las sesiones IP son liberadas y los recursos de red son liberados.

EMPLEO DE CODECS

La naturaleza de las comunicaciones de voz es analógica, mientras que una red de datos tiene una naturaleza digital. Es por ello que se requiere un proceso de conversión entre el mundo analógico al mundo digital, y viceversa. Este trabajo es desarrollado por un CODEC. Existe una gran diversidad de formas para transformar una señal de voz analógica, todas ellas regidas por diferentes estándares. La mayoría de las conversiones se basan en PCM o variantes de ésta

técnica. Además de adecuar la señal al medio de transmisión, un CODEC comprime la secuencia de datos y proporciona cancelación de eco. La compresión de la forma de onda representada, puede permitir un ahorro en el ancho de banda. Otra forma de ahorrar en ancho de banda es al emplear una supresión del silencio, en otras palabras, no enviar paquetes de voz cuando hay silencios en las conversaciones.

Usar la compresión, así como la supresión de silencio, puede provocar un ahorro considerable en ancho de banda. Sin embargo, algunas aplicaciones de VoIP pueden verse fuertemente afectadas por la compresión. Un ejemplo claro de esto, es el impacto en los usuarios de *modems.* Los esquemas de compresión pueden interferir con el funcionamiento de los *modems* confundiendo el esquema de constelación empleado. El resultado puede ser que los *modems* nunca se sincronicen o que tengan un rendimiento pobre.

La salida de los CODECs es una secuencia de datos que se pone en paquetes IP y son transportados a través de la red hasta el destino. Estos destinos deben emplear los mismos estándares, así como los mismos parámetros del CODEC. El resultado de emplear estándares o parámetros diferentes en ambos extremos es una comunicación ininteligible.

Con el objeto de optimizar ancho de banda se idearon CODECs basados en algoritmos predictivos que permiten comprimir la voz eficientizando el ancho de banda requerido. Estos CODECs traen un costo en la comunicación, ya que agregan un retraso de compresión y degradan la voz. En la ITU–T G.114 se habla de un retardo máximo unidireccional de 150 ms.

Se ideó un parámetro denominado MOS (*Mean Opinion Score)* para comparar las calidades de la voz de los distintos CODECs. Este parámetro es puramente cualitativo y subjetivo. Cuanto más parecido a 4.1 resulte este parámetro la voz se asemeja más a la de una red TDM.

Método de Compresión	Bit Rate (kbps)	MOS	Delay de Compresión (ms)
G.711 PCM	64	4.1	0.75
G.726 ADPCM	32	3.85	1
G.728 LD-CELP	16	3.61	3 to 5
G.729 CS-ACELP	8	3.92	10
G.729 x 2 Encodings	8	3.27	10
G.729 x 3 Encodings	8	2.68	10
G.729a CS-ACELP	8	3.7	10
G.723.1 MP-MLQ	6.3	3.9	30
G.723.1 ACELP	5.3	3.65	30

Tabla 2.1. CODECs

Los CODECs son altamente eficientes comprimiendo, pero son muy sensibles a la pérdida de tramas, por ejemplo G.729 bajo su MOS de 3.92 a 1.7 si se pierden 5 tramas consecutivas.

2.2.2 COMPONENTES DE UNA RED VOIP

La mayoría de los componentes de una Red VoIP son muy similares en funcionamiento a aquellos de una red de circuitos conmutados. Las redes VoIP deben ser capaces de realizar las mismas tareas que la PSTN, además de realizar la conexión a la red pública existente. Aunque emplean tecnologías y puntos de vista diferentes, los conceptos elementales de los elementos que generan la PSTN también hacen posible las redes VoIP. Existen tres elementos principales en una red VoIP.

MEDIA GATEWAYS

Son los responsables de la creación y detección de la llamada, de la conversión analógica a digital de la voz y de la generación de los paquetes de voz. Además de esto, pueden tener características opcionales como son: compresión analógica o digital, cancelación de eco, supresión de silencio y recopilación de datos estadísticos.

Son la interfase que el contenido de voz emplea para ser transportado a través de la red IP. Son el origen del tráfico de carga. Típicamente, cada conversación es una sola sesión IP transportada por RTP sobre UDP.

Pueden ser de diferentes formas. Por ejemplo, pueden ser equipos de telecomunicaciones dedicados, o simplemente una PC corriendo software VoIP. Los principales tipos son:

- Troncales, que interactúan entre la PSTN y la red VoIP. Este tipo de *gateways* manejan una gran cantidad de circuitos digitales.
- Residenciales, que proveen una interfase analógica tradicional a una red VoIP. Ejemplos de estos pueden ser cable módems, equipos xDSL así como equipos inalámbricos de banda ancha.
- De acceso, que proveen una interfase analógica tradicional, o una interfase PBX digital a una red VoIP.
- De negocios, que proveen una interfase digital PBX tradicional, o una interfase integrada *soft* PBX a una red VoIP.
- Servidores de acceso a red, que pueden adjuntar un módem a un circuito telefónico y proveer acceso de datos a Internet.
- Teléfonos discretos IP.

CONTROLADORES DE SEÑALIZACIÓN DE *MEDIA GATEWAYS*

Los Controladores de Señalización de *Media Gateways* albergan la señalización y servicios de control que coordinan las funciones de los *Media Gateways*. Pueden considerarse similares a los *Gatekeepers* H.323. Un Controlador tiene la responsabilidad de coordinar algunas o todas las señalizaciones de la llamada, traducciones de números telefónicos, búsqueda del cliente, manejo de recursos, y servicios de señalización a la PSTN. La cantidad de funcionalidad está basada en la particularidad de los productos VoIP empleados.

En una red VoIP escalable, se puede romper la función de un Controlador de Señalización de *Media Gateways* en dos: Controlador de Señalización del *Gateway* y Controlador del *Media Gateway*. En llamadas originadas y terminadas dentro del mismo dominio de la red VoIP, solamente es necesario el Controlador del *Media Gateway* para completar las llamadas. Sin embargo, si una red VoIP frecuentemente se conecta a la red pública, puede emplearse entonces, un

Controlador de Señalización del *Gateway* para conectarse directamente a la red SS7, mientras interactúa también con los elementos de la red VoIP. Este Controlador de Señalización estaría dedicado a la traducción de mensajes y señalización necesaria para vincular la PSTN con la red VoIP.

RED IP

Una red VoIP puede ser vista como un conmutador lógico. Sin embargo, este conmutador lógico es un sistema distribuido, más que sólo una entidad conmutadora. El *backbone* IP provee la conectividad entre los elementos distribuidos. Dependiendo de los protocolos VoIP usados, este sistema, como un todo, es a veces referido como *arquitectura softswitch*.

La infraestructura IP debe asegurar la entrega silenciosa de los paquetes de voz y señalización a los elementos VoIP. Debido a las diferencias entre datos y voz, la red IP debe también tratarlos diferentes. Si la red IP transporta tráfico tanto de voz, como de datos, ésta debe ser capaz de priorizar los diferentes tipos de tráficos.

Existen muchas correlaciones entre los componentes VoIP y de circuitos conmutados, sin embargo, también hay muchas diferencias. Una de ellas es el transporte del tráfico de voz resultante. Las telecomunicaciones basadas en circuitos conmutados, pueden ser clasificadas como redes TDM, que dedican canales, reservando el ancho de banda necesario para interconectar los conmutadores.

Las redes IP son diferentes en cuanto que emplean conmutación de paquetes, y se basa principalmente en disponibilidad estadística. El CoS asegura que a los paquetes de una aplicación específica se les dé prioridad. Esta priorización se requiere para aplicaciones VoIP en tiempo real, para evitar que el tráfico de voz sea afectado por otros flujos de tráfico.

2.3 H.323

El estándar H.323 proporciona la base para la transmisión de voz, datos y vídeo sobre redes no orientadas a conexión y que no ofrecen un grado alto de calidad del servicio, como son las basadas en IP, incluida Internet, de tal manera que las aplicaciones y dispositivos que pertenecen a la Internet puedan interoperar, permitiendo la comunicación entre los usuarios sin necesidad de que éstos se preocupen por la compatibilidad de sus sistemas. La LAN sobre la que los terminales H.323 se comunican puede ser un simple segmento o múltiples segmentos con una topología compleja, lo que puede resultar en un grado variable de rendimiento.

Esta especificación, fijó los estándares para la comunicación de voz y vídeo sobre redes LAN, con cualquier protocolo, que por su propia naturaleza presentan un gran retardo y no garantizan una determinada QoS debido a la infraestructura misma de la red.

El estándar cubre el control de la llamada, gestión de la información y ancho de banda para una comunicación punto a punto y multipunto, dentro de la LAN, y también define interfaces entre la LAN y otras redes externas, como puede ser la Red Digital de Servicios Integrados (ISDN – *Integrated Services Digital Network*). Dentro de sus características están una serie de especificaciones para videoconferencia sobre distintos tipos de redes, que incluyen desde la H.320 a la H.324, implementadas para la ISDN y la PSTN, respectivamente.

2.3.1 COMPONENTES DE H.323

Este estándar define un amplio conjunto de características y funciones. Algunas son necesarias y otras opcionales. El H.323 define mucho más que los terminales. El estándar define los siguientes componentes más relevantes:

ENTIDAD
Cualquier componente que cumpla con el estándar H.323.

EXTREMO
Componente de la red que puede enviar y recibir llamadas. Puede generar y/o recibir secuencias de información.

TERMINAL
Extremo de la red que proporciona comunicaciones bidireccionales en tiempo real con otro terminal H.323, *Gateway* o Unidad de Control Multipunto (MCU). Esta comunicación consta de señales de control, indicaciones, audio, imagen en color en movimiento y/o datos entre los dos terminales. Puede proporcionar sólo voz, voz y datos, voz y vídeo, o voz, datos y vídeo.

GATEKEEPER
Entidad que proporciona la traducción de direcciones y el control de acceso a la red de los terminales H.323, G*ateways* y MCUs. El *Gatekeeper* puede también ofrecer otros servicios tales como gestión del ancho de banda y localización de los *Gateways*.

El *Gatekeeper* realiza dos funciones de control de llamadas que preservan la integridad de la red corporativa de datos. La primera es la traducción de direcciones de los terminales de la LAN a las correspondientes IP o IPX, tal y como se describe en la especificación RAS. La segunda es la gestión del ancho de banda, fijando el número de conferencias que pueden estar dándose simultáneamente en la LAN y rechazando las nuevas peticiones por encima del nivel establecido, de manera tal que se garantice ancho de banda suficiente para las aplicaciones de datos sobre la LAN. El *Gatekeeper* proporciona todas las funciones anteriores para los terminales que están registrados dentro de la denominada Zona de Control H.323.

GATEWAY
Extremo que proporciona comunicaciones bidireccionales en tiempo real entre terminales H.323 en la red IP y otros terminales o *Gateways* en una red conmutada. En general, el propósito del *Gateway* es reflejar transparentemente las características de un extremo en la red IP a otro en una red conmutada y viceversa.

UNIDAD DE CONTROL MULTIPUNTO (MCU – *MULTIPOINT CONTROL UNIT*)
Está diseñada para soportar la conferencia entre tres o más puntos, bajo el estándar H.323, llevando la negociación entre terminales para determinar las capacidades comunes para el proceso de audio y vídeo y controlar la multidifusión.

2.3.2 PROTOCOLOS DE H.323

Debido a la existencia del estándar H.323 del ITU-T, que cubría la mayor parte de las necesidades para la integración de la voz, se decidió que el H.323 fuera la base del estándar VoIP. De este modo, VoIP debe considerarse como una clarificación de H.323, de tal forma que en caso de conflicto, y a fin de evitar divergencias entre los estándares, se decidió que H.323 tendría prioridad sobre el VoIP. VoIP tiene como principal objetivo asegurar la interoperabilidad entre equipos de diferentes fabricantes, fijando aspectos tales como la supresión de silencios, codificación de la voz y direccionamiento, y estableciendo nuevos elementos para permitir la conectividad con la infraestructura telefónica tradicional. Estos elementos se refieren básicamente a los servicios de directorio y a la transmisión de señalización por tonos multifrecuencia (DTMF – *Digital Tone Mutli Frecuency*).

El H.323 comprende a su vez una serie de estándares y se apoya en una serie de protocolos que cubren los distintos aspectos de la comunicación.

DIRECCIONAMIENTO

RAS (*Registration, Admission and Status*): Protocolo de comunicaciones que permite a una estación H.323 localizar otra estación H.323 a través del *Gatekeeper*.

DNS: Resolución de nombres en direcciones IP con el mismo fin que el protocolo RAS pero a través de un servidor DNS.

SEÑALIZACIÓN

Q.931: Señalización inicial de llamada

H.225: Control de llamada señalización, registro y admisión, y paquetización/sincronización del flujo de voz

H.245: Protocolo de control para especificar mensajes de apertura y cierre de canales para voz

COMPRESIÓN DE VOZ

Requeridos: G.711 y G.723

Opcionales: G.728, G.729 y G.722

TRANSMISIÓN DE VOZ

UDP: La transmisión se realiza sobre paquetes UDP, pues aunque UDP no ofrece integridad en los datos, el aprovechamiento del ancho de banda es mayor que con TCP.

RTP: Maneja los aspectos relativos a la temporización, marcando los paquetes UDP con la información necesaria para la correcta entrega de los mismos en recepción.

CONTROL DE LA TRANSMISIÓN:

RTCP: Se utiliza principalmente para detectar situaciones de congestión de la red y tomar, en su caso, acciones correctoras.

2.3.3 CARACTERÍSTICAS DE LOS TERMINALES H.323

Los equipos terminales H.323 presentan las siguientes características principales:

- **CODEC de audio:** Para ser capaces de transmitir y recibir ley A y ley μ. Opcionalmente, puede ser capaz de codificar y decodificar señales vocales. El terminal H.323 tiene capacidad de enviar más de un canal de audio al mismo tiempo, por ejemplo, para hacer posible la difusión de 2 idiomas.

- **CODEC de video:** En los terminales H.323 es opcional.

- **Canal de datos:** Uno o más canales de datos son opcionales. Pueden ser unidireccionales o bidireccionales.

- **Retardo en el trayecto de recepción:** Incluye el retardo añadido a las tramas para mantener la sincronización, y tener en cuenta la fluctuación de las llegadas de paquetes. No suele usarse en la transmisión sino en recepción, para añadir el retardo necesario en el trayecto de audio para, por ejemplo, lograr la sincronización con el movimiento de los labios en una videoconferencia.

- **Unidad de control del sistema:** Proporciona la señalización necesaria para el funcionamiento adecuado del terminal. Está formada por tres bloques principales:

  Control H.245: Se utiliza para llevar mensajes de control extremo a extremo que rige el modo de funcionamiento de la entidad H.323. Se ocupa de negociar el ancho de banda, de la apertura y cierre de los canales lógicos y de los mensajes de control de flujo. En cada llamada, se puede transmitir cualquier número de canales lógicos de cada tipo de medio (audio, video, datos) pero sólo existirá un canal lógico de control, el canal lógico 0.

  Señalización de la llamada: Se utiliza para llevar mensajes de establecimiento y finalización de la llamada entre 2 puntos extremos H.323. El canal de señalización de llamada es independiente del canal de control H.245. Los procedimientos de apertura y cierre de canal lógico no se utilizan para establecer el canal de señalización. Se abre antes del establecimiento del canal de control H.245 y de cualquier otro canal lógico. Puede establecerse de terminal a terminal o de terminal a *Gatekeeper*.

  Control RAS: Se utiliza para llevar a cabo procedimientos de registro, admisión, situación y cambio de ancho de banda entre puntos extremos y el *gatekeeper*. El canal de señalización RAS es independiente del canal de señalización de llamada, y del canal de control H.245. Los procedimientos de apertura de canal lógico H.245 no se utilizan para establecer el canal de señalización RAS. El canal de señalización RAS se abre antes de que se establezca cualquier otro canal entre puntos extremos H.323.

- **Capa H.225:** Se encarga de dar formato a las tramas de video, audio, datos y control transmitidas en mensajes de salida hacia la interfaz de red y de recuperarlos de los mensajes que han sido introducidos desde la interfaz de red. Además lleva acabo también la alineación de trama, la numeración secuencial y la detección/corrección de errores.

- **Interfaz de red de paquetes:** Es específica en cada implementación. Debe proveer los servicios descritos en la recomendación H.225. Esto significa que el

servicio extremo a extremo fiable (TCP) es obligatorio para el canal de control H.245, los canales de datos y el canal de señalización de llamada. El servicio de extremo a extremo no fiable (UDP, IPX) es obligatorio para los canales de audio, los canales de video y el canal de RAS. Estos servicios pueden ser dúplex o simplex y de *unicast* o *multicast* dependiendo de la aplicación, las capacidades de los terminales y la configuración de la red.

2.3.4 CARACTERÍSTICAS DE LOS *GATEKEEPERS*

Elemento opcional en la comunicación entre terminales H.323. No obstante, es el elemento más importante de una red H.323. Actúan como punto central de todas las llamadas dentro de una zona y proporcionan servicios a los terminales registrados y control de las llamadas. De alguna forma, el *Gatekeeper* H.323 actúa como un conmutador virtual.

A la conjunto de todos los terminales, *Gateways* y MCU's gestionados por un *Gatekeeper* se la conoce como Zona H.323.

Una característica opcional, pero valiosa de los G*atekeepers* es la habilidad para enrutar llamadas. Si se enruta la llamada por un *Gatekeeper*, esta puede ser controlada más efectivamente. Los proveedores de servicio necesitan esta característica para facturar por las llamadas realizadas a través de su red. Este servicio también puede ser usado para re-enrutar una llamada a otro terminal en caso de estar no disponible el llamado.

Además, con esta característica un *Gatekeeper* puede tomar decisiones que involucren el balance entre varios *Gateways*. Por ejemplo, si una llamada es enrutada por un G*atekeeper*, ese *Gatekeeper* puede re-enrutar la llamada a uno de varios *Gateways* basándose en alguna lógica de enrutamiento propietaria.

Mientras que un *Gatekeeper* está lógicamente separado de los extremos de una conferencia H.323, los fabricantes pueden elegir incorporar la funcionalidad del *Gatekeeper* dentro de la implementación física de *Gateways* y MCU's.

Los *Gatekeepers* juegan también un rol en las conexiones multipunto. Para soportar conferencias multipunto, los usuarios podrían emplear un *Gatekeeper* para recibir los canales de control H.245 desde dos terminales en una conferencia punto-punto. Cuando la conferencia cambia a multipunto, el *Gatekeeper* puede redireccionar el Canal de Control H.245 a un controlador multipunto. El *Gatekeeper* no necesita procesar la señalización H.245, sólo necesita pasarla entre los terminales o entre los terminales y el controlador multipunto.

Las redes que posean un *Gateway* pueden también tener un *Gatekeeper* para traducir llamadas entrantes E.164 (números de teléfono convencionales) a direcciones de transporte. Debido a que una Zona está definida por su *Gatekeeper*, las entidades H.323 que contengan un *Gatekeeper* interno necesitan de un mecanismo para desactivar su funcionamiento. Cuando hay varias entidades H.323 que contiene un *Gatekeeper* dentro de la red, las entidades pueden ser configuradas para estar en la misma Zona.

Existen dos formas para que un terminal se registre en un G*atekeeper*, sabiendo su IP y enviando entonces un mensaje de registro *unicast* a esta dirección o bien enviando un mensaje *multicast* de descubrimiento del *Gatekeeper* (GRQ).

Funciones obligatorias:

- **Traducción de Direcciones:** Traducción de alias a direcciones de transporte, usando para ello una tabla que es modificada con mensajes de *Registration* u otros métodos.

- **Control de Admisión:** El *Gatekeeper* debe autorizar el acceso a la red usando mensajes H.225.0 *Admission Request* (ARQ)/*Admission Confirm* (ACF)/*Admission Reject* (ARJ). Esto puede basarse en autorización de llamada, ancho de banda, o algún otro criterio que es dejado al fabricante. También puede ser una función nula que admita todas las peticiones.

- **Control de Ancho de Banda:** El *Gatekeeper* debe soportar mensajes *Bandwidth Request* (BRQ)/*Bandwidth Reject* (BRJ)/*Bandwidth Confirm* (BCF). Esto puede usarse para gestión del ancho de banda. También se puede aceptar todas las peticiones de ancho de banda.

- **Gestión de Zona:** El *Gatekeeper* debería suministrar la funciones anteriores a: todos los terminales, MCU's y *Gateways* que se encuentren registrados en su Zona de control.

Funciones opcionales:

- **Señalización de control de llamada:** El *Gatekeeper* puede elegir completar la señalización de llamada con los extremos y procesar la señalización de llamada el mismo. Alternativamente, puede elegir que los extremos conecten directamente sus señalizaciones de llamada. De esta manera el Gatekeeper puede evitar gestionar las señales de control H.225.0.

- **Autorización de llamada:** El *Gatekeeper* puede rechazar una llamada desde un terminal. Las razones para rechazar la llamada pueden ser, acceso restringido desde o hacia un terminal particular o *Gateway*, y acceso restringido durante un periodo de tiempo.

- **Gestión de llamada:** El Gatekeeper puede mantener una lista de las llamadas en curso, esta información puede ser usada para indicar si un terminal está ocupado o para dar información a la función de gestión de ancho de banda.

- **Otros:** estructura de datos de información para la gestión, reserva de ancho de banda y servicios de directorio.

2.3.5 CARACTERÍSTICAS DE LOS *GATEWAYS*

El *Gateway* es un elemento opcional de una conferencia H.323. Es necesario solo si necesitamos comunicar con un terminal que está en otra red. Los *Gateways* proporcionan muchos servicios, el más común es la traducción entre formatos de transmisión y entre procedimientos de comunicación. Además el *Gateway* también traduce entre los CODECs de video y audio usados en ambas redes y procesa la configuración de la llamada y limpieza de ambos lados de la comunicación.

El *Gateway* es un tipo particular de terminal y es una entidad llamable (tiene una dirección).

En general, el propósito del *Gateway* es reflejar las características del terminal en la red basada en paquetes al terminal en la PSTN y al contrario.

Las principales aplicaciones de los *Gateways* son:

- Establece enlaces con terminales telefónicos analógicos conectados a la PSTN
- Establecer enlaces con terminales remotos que cumplen H.320 sobre redes ISDN
- Establecer enlaces con terminales remotos que cumple H.324 sobre la PSTN

Los *Gateways* no se necesitan si las conexiones son entre redes basadas en paquetes.

Muchas funciones del *Gateway* son dejadas al diseñador. Por ejemplo, el número de terminales H.323 que pueden comunicar a través del *Gateway* no es asunto de estandarización. Sucede lo mismo con el número de conexiones a la PSTN, el número de conferencias individuales soportadas, las funciones de conversión de audio, video o datos, y la inclusión de funciones multipuntos. Debido a la incorporación de los *Gateways* a la especificación H.323, la ITU posicionó H.323 como el pegamento que junta todos los terminales para conferencias funcionando juntos.

2.3.6 CARACTERÍSTICAS DE LAS UNIDADES DE CONTROL MULTIPUNTO (MCU)

La MCU soporta conferencias entre tres o más extremos.

En terminología H.323, el MCU se compone de:
Controlador Multipunto (MC) que es obligatorio, y
Cero o varios Procesadores Multipunto (MP).

El MC gestiona las negociaciones H.245 entre todos los terminales para determinar las capacidades comunes para el procesado de audio y video. El MC también controla los recursos de la conferencia para determinar cuales de los flujos, si hay alguno, serán *multicast*. Las capacidades son enviadas por el MC a todos los extremos en la conferencia indicando los modos en los que pueden transmitir. El conjunto de capacidades puede variar como resultado de la incorporación o salida de terminales de la conferencia.

El MC no trata directamente con ningún flujo de datos, audio o video. Esto se lo deja al MP, este mezcla, conmuta y procesa audio, video y/o bits de datos. Las capacidades del MC y MP pueden estar implementadas en un componente dedicado o ser parte de otros componentes H.323, en concreto puede ser parte de un *Gatekeeper*, un *Gateway*, un terminal o una MCU.

CONFERENCIAS MULTIPUNTO

Existe una gran variedad de métodos de gestionar las conferencias multipunto. La Recomendación hace uso de los conceptos de conferencia centralizada y descentralizada. Las conferencias centralizadas requieren de una MCU.

Todos los terminales envían audio, video, datos y flujos de control a la MCU en un comportamiento punto a punto.

El MC gestiona de forma centralizada la conferencia usando las funciones de control H.245 que también definen las capacidades de cada terminal. El MP mezcla el audio, distribuye los datos y mezcla o conmuta el video y envía los resultados en flujos de vuelta a cada terminal participante.

En conferencias multipunto descentralizadas se puede hacer uso de tecnología *multicast*. Los terminales H.323 participantes envían audio y video a otros terminales participantes sin enviar los datos a una MCU. Sin embargo el control de los datos multipunto sigue siendo procesado de forma centralizada por la MCU, y la información del canal de control H.245 sigue siendo transmitida de modo unicast a un MC.

Son los terminales que reciben múltiples flujos de audio y video los responsables de procesarlos. Los terminales usan los canales de control H.245 para indicar a un MC cuantos flujos simultáneos de video y audio son capaces de decodificar. El número de capacidades simultáneas de un terminal no limita el número de flujos de audio y video que son enviados por *multicast* en una conferencia.

Las conferencias multipunto híbridas usan una combinación de características de las centralizadas y descentralizadas. Las señalizaciones son procesadas a través de mensajes punto a punto enviados a la MCU. Las restantes señales (audio o video) son enviadas a los participantes a través de *multicast*.

Una ventaja de las conferencias centralizadas es que todos los terminales soportan comunicaciones punto a punto. La MCU puede sacar varios flujos *unicast* a los participantes y no se requiere ninguna capacidad de la red especial.

También es posible que la MCU reciba varios flujos *unicast*, mezcle el audio, y conmute el video, y saque un flujo *multicast*, conservando de esta manera el ancho de banda de la red.

Multicast hace más eficiente el uso del ancho de banda de la red, pero supone una más alta carga computacional en los terminales que tienen que mezclar y conmutar entre los flujos de audio y video que reciben. Además, el soporte *multicast* es necesario en elementos de la red como *routers* y *switches*.

2.3.7 TRANSPORTE DE VOIP

H.323 utiliza comunicaciones seguras e inseguras. En el caso de señales de control y datos su transporte debe ser seguro. No se pueden perder partes de la información y se debe recuperar en el orden en que fue enviada. Para ello se usa TCP que garantiza que la información se recibe libre de errores y en la secuencia correcta. Sin embargo, puede existir un retardo considerable en la recepción. Este protocolo es válido para el control H.245, el canal de datos T.120 y el canal de señalización de llamadas. Sin embargo, carece de sentido para la transmisión de audio y vídeo en tiempo real, puesto que si un paquete de información llega tarde ya no es útil para el receptor. Por ello, en audio y vídeo se usa un protocolo de transmisión inseguro que a cambio ofrece una transmisión más eficiente, que en el IP es el UDP.

UDP usa un número mínimo de bits extras de protocolo ya que se establece un circuito virtual en la red desde el emisor al receptor. De esta forma todos los paquetes de información seguirán la misma ruta y por tanto será la propia red la que se encargará de que se entreguen en el mismo orden en que fueron introducidos. H.323 utiliza UDP para la información de audio, vídeo y el canal de RAS. UDP asegura el mejor esfuerzo para entregar los paquetes de información al receptor, pero es posible que existan pérdidas de paquetes, recepción de paquetes duplicados, etc.

Se usa un protocolo de tiempo real (RTP) para gestionar el audio y el vídeo. De esta forma se añade una cabecera a cada paquete que contiene el código de tiempo y un número de secuencia. El receptor incorpora un *buffer* y a partir de la información de cabecera reordena los paquetes, sincroniza el sonido con el vídeo, elimina los paquetes duplicados y realiza una reproducción continua.

Dado que resulta crítico disponer en una red IP del ancho de banda necesario para una aplicación multimedia, se usa el protocolo RSVP, que permite a un receptor pedir una cantidad de ancho de banda determinada y recibir una respuesta indicando si es posible asegurar la petición.

2.3.8 CARACTERÍSTICAS DE LOS TELÉFONOS IP DE SOFTWARE

Tanto la voz, como las llamadas telefónicas, pueden ser vistas como una de las muchas aplicaciones de una red IP, mediante un software usado como soporte de la aplicación y de la interfaz de la red.

2.3.8.1 MÓDULO DE PROCESAMIENTO DE VOZ

Es el encargado de preparar las muestras de voz para la transmisión de los paquetes a través de la red IP y debe incluir los siguientes elementos y funciones:

- **Interfase PCM:** Recibe muestras PCM de la interfase digital y las envía a los módulos DSP (*Digital Signal Processor)* apropiados para su procesamiento, las muestras PCM recibidas de los diversos DSP's y que ya han sido procesadas se envían a la interfase digital, se ejecutan continuos re-muestreos de fase de las salidas muestreadas que van a la interfase digital para evitar los deslizamientos.

- **Generador de Tonos:** Genera tonos duales de multifrecuencia (DTMF – *Dual Tone Multi Frecuency*) y tonos de llamada en progreso bajo el comando del *host* (teléfono, fax, módems, PBX o conmutadores telefónicos) y se puede configurar para tonos internacionales o americanos.

- **Cancelación de Eco:** Hace cancelación de eco según la norma G.165 en las señales muestreadas de los puertos *full duplex*.

- **Detector de Activación de Voz:** Cuando no se detecta actividad por un periodo de tiempo configurado, el software informa que el paquete esta en silencio, resultando en ahorro de ancho de banda. Este software también mide las características libres de ruido de una interfase telefónica. Reporta esta información al protocolo de paquetes de voz para transmitirla al extremo remoto y así generar ruido cuando no haya voz presente.

- **Detector de Tonos:** Detecta la recepción de DTMF y hace discriminación entre voz y fax. Los tonos detectados se reportan al host para que se activen las funciones apropiadas de voz o fax.

- **Software de Codificadores de Voz:** Comprime los datos de voz para que sean transmitidos en paquetes de datos. Es capaz de hacer varias compresiones en una arquitectura modular.

- **Software de Fax:** Desempeña la función de transmisión de fax al demodular datos PCM, resumir la información relevante y empacar los datos fax analizados en marcos para transmitirlos por la red de paquetes.

- **Unidad de Salida:** Esto hace *buffer* en los paquetes de voz recibidos de la red de paquetes y los envía al codificador de voz para su transmisión.

- **Protocolo de Paquetes de Voz:** Encapsula los datos de voz y fax comprimidos para la transmisión extremo a extremo, entre dos puertos, por el *backbone* de la red.

- **Software de la Interfase de Control:** Coordina el intercambio de información de control y monitoreo entre el DSP y el *host* vía un mecanismo de correo de voz. La información intercambiada incluye la configuración de los datos y el estatus de reportes.

- **Ambiente de Portabilidad en Tiempo Real:** Provee un ambiente de operación para el software que reside en el DSP. Esto hace funciones de sincronización, tareas de gestión: gestión de memoria, y gestión de tiempos.

2.3.8.2 MÓDULO DE SEÑALIZACIÓN

Actúa como un *Gateway* de señalización, permitiendo que las llamadas sean establecidas a través de la red telefónica convencional y debe presentar los siguientes elementos:

- **Software de la Unidad de Interfase Telefónica:** Monitorea periódicamente las interfases de señalización del modulo y proporciona recolección rotatoria y no salteada de los dígitos para la interfase.

- **Unidad de Control de Red:** Convierte la información de señalización telefónica en un formato compatible con el protocolo de señalización para el establecimiento de la sesión de paquetes de voz.

- **Unidad de Traducción de Dirección:** Convierte la dirección de discado E.164 a una dirección que pueda ser usada por la red de paquetes.

- **Controlador de la Interfase DSP:** Transmite información de control entre el microprocesador del *host* y los DSP's.

2.3.8.3 MÓDULO DE GESTIÓN DE RED

Es el encargado de definir la forma de acceso a la red, ya sea IP, ATM o Frame Relay y cuenta con:

- **Pila de Señalización IP:** Involucra el control de llamada H.323 y el software de transporte, incluyendo H.225, H.245, RTP, TCP, IP, UDP.

- **Pila de Protocolos de Señalización ATM:** Para el establecimiento, mantenimiento y liberación de la conmutación punto a punto y punto a multipunto de circuitos virtuales utiliza protocolos de encapsulación de voz sobre ATM, pilas de protocolos de señalización de interfases usuario-red.

- **Pila de Protocolos Frame-Relay:** Incluye protocolos de encapsulación de voz sobre Frame-Relay, soporte de SVC y PVC (circuitos virtuales permanentes), interfase de gestión local (LMI), gestión de congestión, monitoreo de trafico, aplicación de rangos de información comprometida (CIR).

2.3.8.4 MÓDULO DE PROTOCOLO DE RED

Éste módulo se encarga principalmente de:

- **Interfase física con el teléfono final.**

- **Servicio de canal de voz para procesar señalización en un canal de voz y conversión entre muestras PCM y paquetes de voz comprimidos.**

2.3.9 FASES DE COMUNICACIÓN MEDIANTE PROTOCOLOS DE H.323

Las entidades H.323 establecen conexiones en diferentes fases. Si consideramos un escenario en el cual exista un *Gatekeeper*, la conexión entre dos terminales dependientes de este *Gatekeeper* sigue los siguientes pasos:

Fase A: Establecimiento de Llamada. La entidad origen, envía mensajes RAS solicitando la identificación del usuario destino utilizando un mensaje ARQ. El *Gatekeeper* aceptará la llamada y enviará al terminal llamante un mensaje de confirmación (ACF) o bien rechazará la llamada (ARJ). En caso positivo, la entidad origen establecerá una conexión TCP con el terminal destino para establecer el canal de señalización H.225.0. Para ello utilizará la información (dirección IP y puerto) recibidos del *Gatekeeper* a través del mensaje ACF. La entidad origen al recibir dicha conexión contactará con su *Gatekeeper* a través del canal RAS solicitando permiso para poder contestar (ARQ). En caso positivo (ACF), la entidad origen aceptara la conexión y a través de dicho canal (H.225.0) enviará la dirección (dirección IP y puerto) donde establecer el canal H.245 para negociación de parámetros y control de la comunicación.

Una vez obtenida esta información, la conexión puede ser finalizada, ya que no es necesario intercambiar más parámetros a través de este canal.

Fase B: **Intercambio de capacidades. (H.245)**. Establecido el canal H.245 a través de una nueva conexión TCP, las entidades origen y destino determinarán los parámetros de la comunicación: CODECs a utilizar, numero de conexiones y direcciones a utilizar, puertos, numero de muestras por trama, función maestro-esclavo, etc, lo que les permitirá establecer canales para la transmisión de medios (audio, vídeo y datos). Esta conexión debe permanecer mientras intercambien información los terminales y les permite modificar parámetros.

Fase C: Intercambio de información audiovisual. En este punto, ambos terminales establecen canales de información a través de la arquitectura RTP/UDP/IP para el transporte de medios, así como canales de control a través de la arquitectura RTCP/UDP/IP para los canales de realimentación, al objeto de controlar la calidad de los flujos de información recibida por el otro extremo de la comunicación.

Fase D: Terminación de llamada. Tras el intercambio de información audiovisual y al objeto de finalizar la llamada, las entidades H.323 deben informarse a través del canal H.245 mediante el envío de las primitivas de finalización de llamadas, que finalizará con el envío de la primitiva *EndSessionCommand* que provocará el cierre del canal H.245. Además deberán informar al *Gatekeeper* mediante el envío del mensaje RAS *Disengage Request* (DRQ) que permitirá al *Gatekeeper* liberar recursos y proporcionar información de tarificación entre otras.

Sobre este escenario básico existen múltiples variantes en función de la presencia o no del *Gatekeeper* y del rol que el mismo realice. El *Gatekeeper* podría encaminar la información de control (H.225.0 y H.245) o no en función del modelo elegido (Directo o Indirecto).

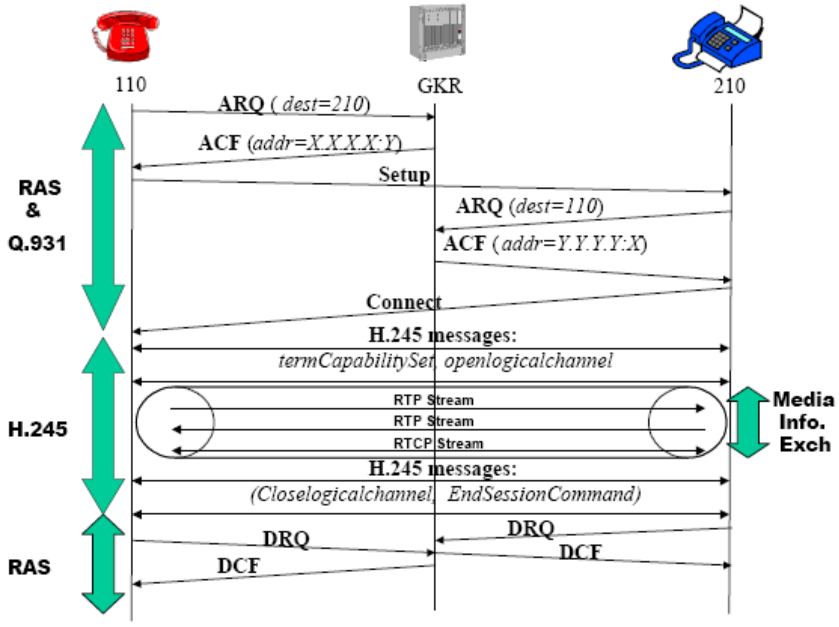


Figura 2.4. Fases de Comunicación del Protocolo H.323

2.3.9.1 PROCESO DE COMUNICACIÓN H.323

En H.323 se encuentran 3 tipos de mensajes de señalización diferentes:

H.245: se describen estos mensajes en forma de texto concatenado en letras tipo *bold* (por ejemplo se menciona el mensaje: **maximumDelayJitter**).
RAS: se representa mediante 3 letras (por ejemplo ARQ).
H.225/Q.931: representado en una o dos palabras con la primer letra en mayúsculas (ejemplo: Call Proceeding). Es usado para encapsular los mensajes H.245 de señalización entre terminales y originalmente fue diseñado como protocolo DSS1 en capa 3/7 para los accesos ISDN.

Fase A: Dentro de esta fase de la comunicación, se realiza un intercambio de mensajes para mantener activa la conexión entre los *Gateways* y el *Gatekeeper*, y presenta diferentes procesos:

***Discovery*:** Este primer proceso es en el cual el *Gateway* determina cual es el *Gatekeeper* que atiende a la red en ese momento. El mensaje desde el *Gateway* es del tipo multicast y se denomina **GRQ** (*Gatekeeper Request*). El *Gatekeeper* responde con la aceptación **GCF** (*GK Confirmation*) o rechazo **GRJ** (*GK Reject*). El *Gatekeeper* puede indicar un *Gatekeeper* alternativo mediante mensajes **alternateGatekeeper**. Si no se está en condiciones de procesar la petición, se puede enviar un mensaje **RIP** (*Requst in Progress*) para indicar que se está procesando la petición.

***Registration*:** El *Gateway* informa de sus direcciones de transporte y alias mediante **RRQ** (*Registration Request*) y el *Gatekeeper* responde con **RCF** (*Registration Confirmation*) o **RRJ** (*Registration Reject*). El RRQ se emite en forma periódica. El proceso de registro tiene un tiempo de duración (expresado en segundos) para lo cual se utiliza el mensaje **timeToLive**. El terminal o el *Gatekeeper* puede cancelar el proceso de registro mediante el mensaje **URQ** (*Unregister Request*) al cual le corresponde la confirmación **URF** (*Unregister Confirmation*).

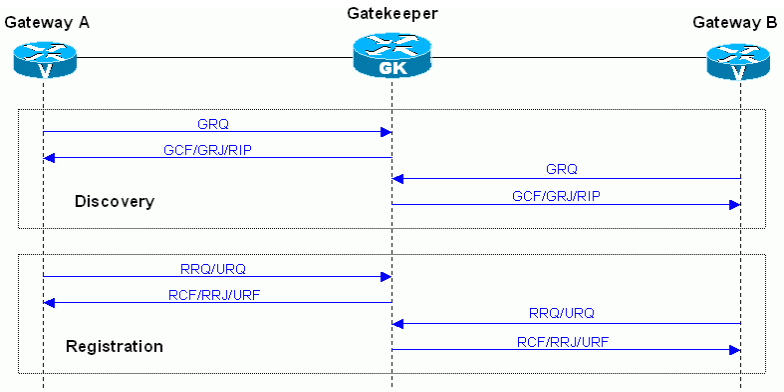


Figura 2.5. Fase A. *Discovery* y *Registration*

***Location*:** Un *Gateway* o *Gatekeeper* que tiene un alias para un *Gateway* y quiere determinar su información de contacto, puede emitir el mensaje de requerimiento de localización **LRQ** (*Location Request*). Al cual le corresponde la confirmación **LCF** (*Location Confirmation*) con la información requerida. La dirección puede ser del tipo E.164 si se trata de un *Gatekeeper* fuera de la red.

De existir varios *Gatekeepers* se disponen de mensajes para intercomunicación, por ejemplo, **LRQ** para *Locate Request* y **LCF** para *Locate Confirm*.

***Status*:** Se trata de un mensaje periódico (mayor a 10 segundos) que emite el *Gatekeeper* al terminal para determinar el estado y requerir un diagnóstico. Se trata de los mensajes **IRQ** (*Information Request*) e **IRR** (*Information Response*). La habilitación se realiza mediante **willRespondToIRR** enviado en el mensaje RCF o ACF.

***Admission*:** El proceso se inicia cuando desde la PSTN se recibe un mensaje de *Setup* para inicio de una llamada entrante en protocolo ISUP. El *Gateway* responde a la PSTN mediante el mensaje Call Proceesing, para mantener la conexión en espera.

El *Gateway* requiere iniciar una llamada mediante el pedido de admisión desde *Gateway* al *Gatekeeper*. Este mensaje es **ARQ** (*Admissions Request*) y contiene un requerimiento *Call Bandwidth* (en formato Q.931). El *Gatekeeper* puede reducir las características de la solicitud en el mensaje de confirmación **ACF** (*Admissions Confirm*). En el mismo mensaje ARQ se dispone de la funcionalidad **TransportQOS** para habilitar la funcionalidad de reservación de ancho de banda RSVP, para servicios unidireccionales (orientado-al-receptor).

***Setup*:** Una vez admitido el *Gateway B* por el *Gatekeeper* el procedimiento se bifurca en el modo ruteado y no-ruteado.

Modo No-Ruteado. En el modo de operación no-ruteado, el *Gatekeeper* informa al *Gateway B* cual es la dirección IP del *Gateway A* al cual va dirigida la llamada, de acuerdo con la dirección E.164 recibida en el mensaje ARQ. Ahora, el *Gateway B* se comunica con el *Gateway A* que fue indicado por el Gatekeeper y le envía el mensaje *Setup*. Este mensaje (en protocolo Q.931) es respondido mediante el mensaje *Call Proceesing*.

El *Gateway A* se ocupa de registrarse mediante ARQ y recibe desde el *Gatekeeper* el mensaje ACF. Con estas acciones cumplidas, el *Gateway A* se ocupa de informar al usuario de la llamada entrante (corriente de llamada al teléfono) y hacia el *Gateway B* le envía el mensaje de Alerting en Q.931 para indicar el estado de llamada. El *Gateway B* envía el mensaje de Alerting a la PSTN, ahora en formato de protocolo ISUP.

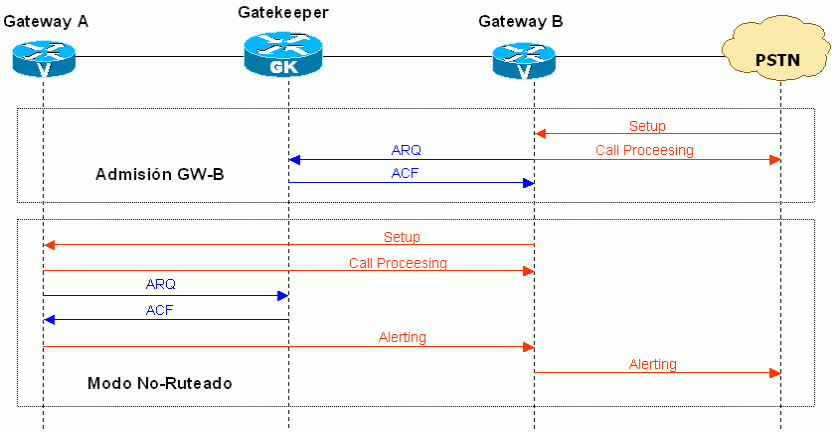


Figura 2.6. Fase A. *Admission* y *Setup* Modo No-ruteado

Modo Ruteado: Para el caso de trabajar con Modo Ruteado, el mensaje de *Setup* entre *Gateway* pasa por el *Gatekeeper*. En el caso No-Ruteado anterior, el *Gatekeeper* se desentiende de la conexión y solo se ocupa de la traslación entre direcciones E.164 e IP. En el modo ruteado el *Gatekeeper* seguirá toda la conexión, de forma que haciendo uso de las funcionalidades de *Softswitch* se podrán ofrecer servicios de valor agregado.

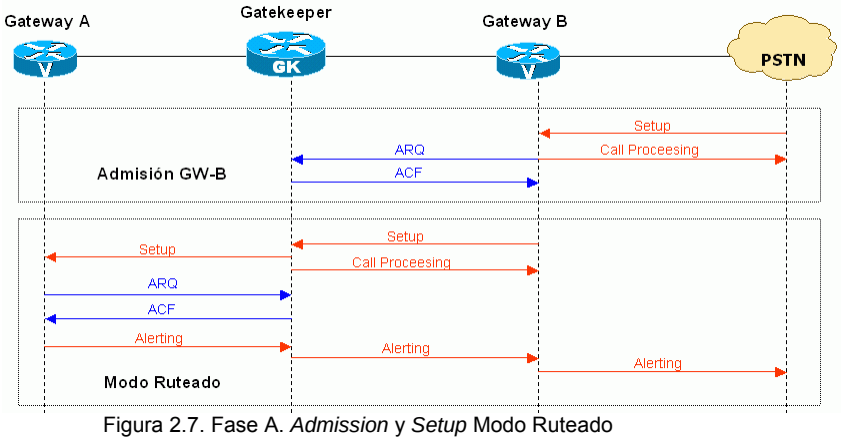


Figura 2.7. Fase A. *Admission* y *Setup* Modo Ruteado

***Connect*:** Cuando el usuario en el *Gateway A* responde se genera el mensaje Q.931 de *Connect*. Este mensaje se emite hacia el *Gatekeeper* (Modo Ruteado), quien hace lo mismo hacia el *Gateway B* y este lo imita hacia la PSTN pero en protocolo ISUP.

Fase B: Dentro de esta fase se establecen las capacidades de los terminales utilizando el protocolo H.245 entre *Gateway*s. Se trata del mensaje **TerminalCapabilitySet** de solicitud y el **TerminalCapabilityAck** de respuesta, que permite determinar capacidad del terminal, tipo de codificador, canal lógico, etc. Finalmente, se envía el mensaje **OpenLogical Channel** para abrir un canal lógico.

Fase C:

Canal Vocal. El canal vocal se transporta sobre los protocolos RTP.

Bandwidth. Durante una conexión el terminal o el *Gatekeeper* pueden requerir el cambio de ancho de banda del canal mediante el mensaje **BCR** (*Bandwidth Change Request*).

Fase D: En la figura se indica la fase de desconexión de la llamada. La misma se realiza con mensajes Release Complete de Q.931 y **DRQ** (*Delete Request*) y **DCF** (*Delete Confirm*) de RAS.

Sobre el paquete Q.931 (H.225) se disponen de distintos tipos de mensajes:

- Mensajes para establecimiento de llamada: *Alerting, Call Proceeding, Connect, Setup, Progress*, etc.
- Mensajes para la fase de información de llamada: *Resume, Suspend, User Information*, etc.
- Mensajes para el cierre de la llamada: *Disconnect, Release, Restart*, etc.
- Mensajes misceláneos: *Segment, Congestion Control, Information, Notify, Status, Status Enquiry*, etc.

Los mensajes manejados en el ámbito de H.245 (durante la fase de comunicación telefónica) son:

- **multimediaSystemControl** para efectuar el control del sistema; las variantes del mensaje son request, response, command and indication.
- otros mensajes de interés son: **masterSlaveDetermination, terminalCapability, MaintenanceLoop, communication Mode, communicationMode, conferenceRequest and Response, terminal-ID.**

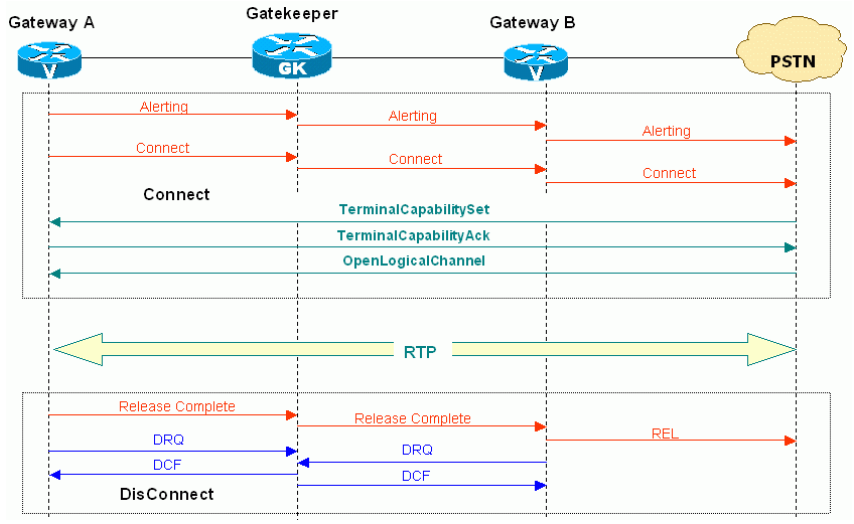


Figura 2.8. Comunicación H.323

2.3.9.1.1 FORMATO DE LOS PAQUETES H.225/245

La estructura del H.225 sigue el estándar Q.931 como se muestra en la figura:

0	1	2	3	4	7
Discriminador Protocolo					
0	0	0	0	Valor de Referencia	
Valor de Referencia de la Llamada					
Tipo de Mensaje					
Elementos de Información					

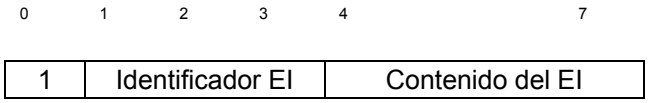
- Discriminador de Protocolo (8 bits): Identifica el mensaje de control de llamada en la interfaz usuario-red.
- Valor de Referencia (4 bits): Indica la longitud del valor de referencia de la llamada.
- Valor de Referencia de la Llamada (8 - 16 bits): Identifica la petición de registro o cancelación en la interfase de la red local del usuario a la cual un mensaje particular aplica.
- Tipo de Mensaje (8 bits): Identifica el tipo de mensaje enviado. Los tipos empleados son:

000 xxxxx	Mensajes para establecimiento de llamada:
00001	ALERTING
00010	CALL PROCEEDING
00111	CONNECT

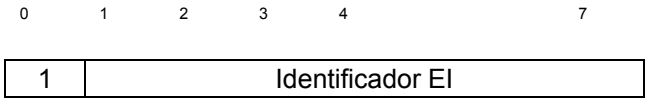
01111	CONNECT KNOWLEDGE
00011	PROGRESS
00101	SETUP
01101	SETUP ACKNOWLEDGE
001 xxxxx	Mensajes para la fase de información de llamada:
00110	RESUME
01110	RESUME ACKNOWLEDGE
00010	RESUME REJECT
00101	SUSPEND
01101	SUSPEND ACKNOWLEDGE
00001	SUSPEND REJECT
00000	USER INFORMATION
010 xxxxx	Mensajes para el cierre de llamadas:
00101	DISCONNECT
01101	RELEASE
11010	RELEASE COMPLETE
00110	RESTART
01110	RESTART ACKNOWLEDGE
011 xxxxx	Mensajes Misceláneos:
00000	SEGMENT
11001	CONGESTION CONTROL
11011	INFORMATION
01110	NOTIFY
11101	STATUS
10101	STATUS ENQUIRY

- Elementos de Información (8 bits): Se definen dos categorías de elementos de información (EI): Elementos de información de byte simple y elementos de información de longitud variable:

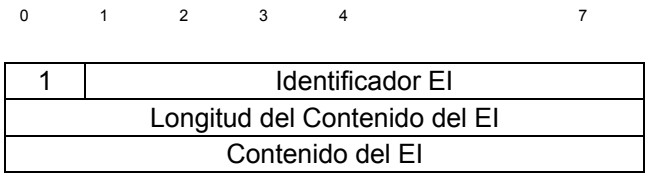
  - Información de *Byte* Simple Tipo I



  - Información de Byte Simple Tipo II



  - Información de Longitud Variable

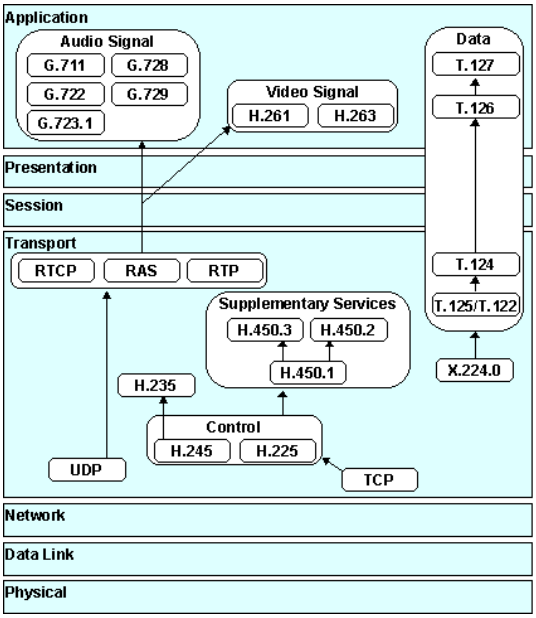


Por otra parte, H.245 es la línea de transmisión de señales no telefónicas. Incluye las capacidades de recepción y transmisión, así como el modo preferido de recepción en el punto final de recepción, señalización lógica del canal y Control e Indicación. Los procedimientos de señalización conocidos se especifican para asegurar comunicaciones audiovisuales y de datos confiables.

Los mensajes H.245 tienen sintaxis ASN.1. Los principales mensajes H.245 son:

- Determinación Maestro Esclavo
- Capacidades del Terminal
- Señalización del Canal Lógico
- Señalización de la Tabla de Multiplexación
- Petición de la Tabla de Multiplexación
- Modo de Petición
- Retraso de Viaje Redondo
- Mantenimiento de Malla
- Modo de Comunicación
- Petición y Respuesta a Conferencia
- Identificación del Terminal
- Control del Sistema Multimedia
- Comandos e Indicaciones

2.3.10 MODELO DE CAPAS Y PROTOCOLOS UTILIZADOS EN VOIP



2.4 PROTOCOLOS DE RED USADOS POR VOIP

A continuación se describen los principales protocolos de red empleados para poder brindar un buen servicio en lo referente a VoIP.

2.4.1 PROTOCOLO ISUP (ISDN *USER PART*)

El protocolo ISUP, es clave en el Sistema de Señalización 7 (SS7). Define los protocolos y procedimientos empleados para configurar, manejar y liberar circuitos troncales que transportan llamadas de voz y de datos a través de la PSTN entre diferentes conmutadores. ISUP se emplea tanto para llamadas ISDN como para las que no lo son. El flujo de una llamada simple empleando ISUP es:

Configuración de Llamada: Cuando una llamada se realiza hacia un número fuera del conmutador, el SSP (*Service Switching Point*) transmite un IAM (*Initial Address Message*) para reservar un circuito inactivo desde el conmutador origen hasta el conmutador destino. El conmutador destino timbra la línea deseada, si es que está disponible, y transmite un mensaje ACM (*Address Complete Message*) al conmutador origen, para indicarle que se ha reservado el punto final remoto del circuito troncal.

El STP (*Signalling Transfer Point*) enruta el ACM al conmutador origen, el cual a su vez, timbra el teléfono origen, y lo conecta al circuito troncal, para completar la llamada desde el punto origen hasta el destino.

Conexión de Llamada: Cuando el usuario destino levanta el teléfono, el conmutador destino finaliza el tono de timbrado y transmite un mensaje ANM (*ANswer Message*) al conmutador origen a través de su STP local. El STP enruta el ANM al conmutador origen, el cual verifica que la línea origen está conectada al circuito troncal reservado, si es así, inicia la tarificación.

Terminación de Llamada: Si el usuario origen cuelga primero, el conmutador origen envía un mensaje REL (*RELease*) y liberar el circuito troncal entre los conmutadores. El STP enruta el REL al conmutador destino. Si por el contrario, el usuario destino cuelga primero, o la línea está ocupada, el conmutador destino envía un REL al conmutador origen, indicando la causa de la liberación, ya sea liberación normal u ocupado. Tras recibir el mensaje REL, el conmutador destino desconecta la troncal de la línea origen, establece el estado de la troncal a “inactivo” y transmite el mensaje RLC (*ReLease Complete*) al conmutador origen para informar la liberación del punto final remoto de la troncal. Una vez que el conmutador origen recibe, o genera el RLC, termina el ciclo de tarificación y establece el estado de la troncal a “inactivo” esperando la siguiente llamada.

2.4.2 PROTOCOLO RTP

El protocolo RTP tiene como objetivo asegurar QoS para servicios del tipo tiempo real. Incluye la identificación de la carga, la numeración secuencial, la medición de tiempo y el reporte de la calidad.

Entre sus funciones se encuentran la memorización de datos, la simulación de distribución interactiva, el control y mediciones de aplicaciones.

RTP trabaja en la capa de Transporte del Modelo OSI y sobre UDP, de forma que posee un *checksum* para detección de errores y la posibilidad de multiplexación de

puertos, para ello se recurre a un doble direccionamiento mediante las direcciones IP y el número de puerto en UDP.

RTP funciona en conjunto con RSVP en la capa de Red del Modelo OSI para la reservación de ancho de banda y proveer QoS Garantizada. La QoS Diferenciada se logra mediante la priorización de tráfico que puede adoptar dos alternativas. En IP se pueden asignar diversas alternativas de prioridad para formar una cola de espera en *routers*.

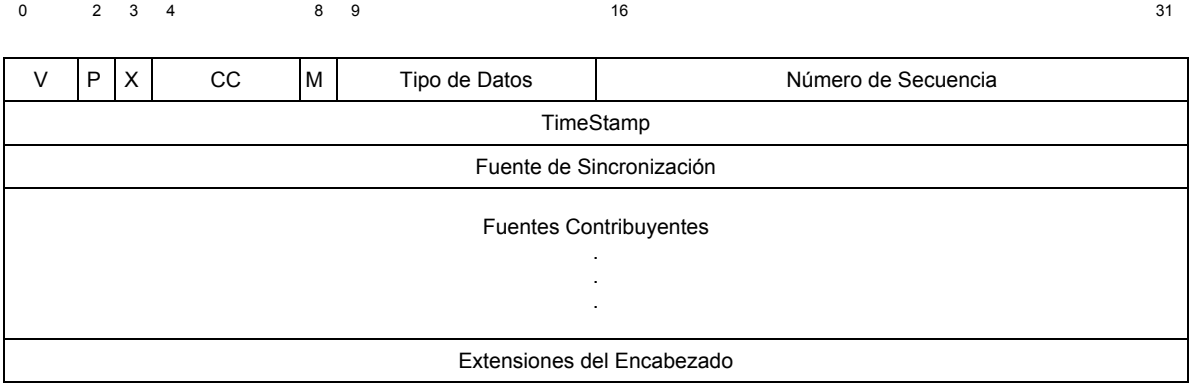
Un algoritmo particular de gestión de prioridad de tráfico es el WFQ (*Weighted Fair Queuing*) que utiliza un modelo de multiplexación TDM para distribuir el ancho de banda entre clientes. Cada cliente ocupa un intervalo de tiempo en un *Round-Robin*.

El ToS en IP puede determinar un ancho de banda específico para el cliente, por ejemplo, un servicio sensible al retardo requiere un ancho de banda superior. En IP además del ToS se pueden utilizar la dirección de origen y destino IP, tipo de protocolo y número de *socket* para asignar una ponderación. En redes que disponen de conmutadores para la capa de Enlace del Modelo OSI, se requiere extender la gestión de la calidad de servicio a dicha capa. Para ello la IEEE ha determinado el ToS sobre IEEE-802.

RTP provee además, transporte para direcciones *unicast* y *multicast*. Por esta razón, también se encuentra involucrado el protocolo IGMP para administrar el servicio *multicast*. El paquete de RTP incluyen un encabezado fijo y el *payload* de datos; RTCP utiliza el encabeza del RTP y ocupa el campo de carga útil.

2.4.2.1 CAMPOS DEL PROTOCOLO RTP

La siguiente figura muestra el encabezado utilizado por el protocolo RTP.



- Version (V, 2 bits). Identifica la versión del protocolo.
- *Padding* (P, 1 bit): Informa que los datos de RTP llevan datos de relleno para completar un bloque de cierto tamaño. El último byte en el mensaje UDP dice de qué tamaño es el relleno.
- Extensión (X, 1 bit): El bit de extensión es utilizado para indicar la presencia de un encabezado de extensión que puede ser definido para una aplicación específica y sigue al encabezado principal. Ese tipo de encabezados son utilizados en raras ocasiones ya que es posible definir un encabezado dentro de los datos como parte de la definición del formato de los datos para una aplicación en particular.

- CC (4 bits): Número de fuentes contribuyentes incluidas en el encabezado de RTP.
- Marcador (M, 1 bit): Este bit es utilizado para indicar el *frame*. Por ejemplo, puede indicar el inicio de una conversación en RTP: el primer *frame*.
- Tipo de Datos (7 bits): Indican qué tipo de dato se está transportando.
- Numero de Secuencia (16 bits): Utilizado para permitir al receptor de un *stream* RTP detectar paquetes perdidos o que lleguen en desorden.
- *Timestamp* (32 bits): Permite al receptor reproducir las muestras en los intervalos de tiempo apropiados y permite que diferentes medios se puedan sincronizar.
- Fuente de Sincronización (32 bits): Identifica de manera única una sola fuente en un *stream* RTP. En una conferencia multimedial, cada emisor escoge un SSRC aleatorio. El identificador de fuente es diferente de la dirección IP o del número de puerto, que permite, por ejemplo, que un nodo con múltiples fuentes (un equipo con varias cámaras) distinga cada una de las fuentes. Cuando un sólo nodo genera diferentes *streams* multimedia (por ejemplo, audio y video al mismo tiempo), no es necesario que utilice el mismo SSRC en cada *stream* ya que RTCP tiene un mecanismo para hacer sincronización intermedia.
- Fuentes Contribuyentes (0 a 15 elementos, cada uno de 32 bits): Es utilizado sólo cuando varios *streams* RTP pasan a través de un mezclador. Un mezclador puede ser utilizado para reducir los requerimientos de ancho de banda para una conferencia recibiendo datos de muchas fuentes y enviando estas como un sólo *stream*. Si hay más de 15 fuentes contribuyentes, sólo 15 pueden ser identificadas.

2.4.2.2 RTP-HC (RTP – *HEADER COMPRESSION*)

Protocolo que permite la compresión del encabezado para mejorar la eficiencia del enlace en paquetes de corta longitud en la carga útil. Se trata de reducir los 40 Bytes de encabezado en RTP/UDP/IP a una fracción de 2 a 5 Bytes, eliminando aquellos que se repiten en todos los paquetes. Como los servicios de tiempo-real generalmente trabajan con paquetes pequeños y generados en forma periódica se procede a formar un encabezado de longitud reducida que mejore la eficiencia de la red.

2.4.3 PROTOCOLO DE CONTROL RTCP

Este protocolo permite completar a RTP facilitando la comunicación entre extremos para intercambiar datos y monitorear de esta forma la calidad de servicio y obtener información acerca de los participantes en la sesión. RTCP se fundamenta en la transmisión periódica de paquetes de control a todos los participantes en la sesión usando el mismo mecanismo de RTP de distribución de paquetes de datos. El protocolo UDP dispone de distintas puertos como mecanismo de identificación de protocolos.

La función primordial de RTCP es la de proveer una realimentación de la calidad de servicio. Se relaciona con el control de congestión y flujo de datos.

CAPÍTULO 3
PROTOCOLO SIP (*SESSION INITIATION PROTOCOL*)

3.1. HISTORIA

Actualmente se está desarrollando e iniciando el despliegue de un conjunto de protocolos y tecnologías de red con las que se pretende sentar las bases necesarias para que la comunicación multimedia en tiempo real a través de Internet sea tan accesible como la comunicación de texto y datos. Además, se pretende que estos nuevos sistemas puedan interoperar con el sistema de conferencia en tiempo real más extendido: la red telefónica.

Para lograr un sistema de telecomunicación universal en el que tengan cabida contenidos tan diversos como correo-e, vídeo, voz o fax, entre otros muchos, son necesarios avances en diversos campos. Nos centraremos en la arquitectura y los protocolos propuestos por el IETF, dejando al margen tecnologías como las relacionadas con los formatos multimedia o las tecnologías e infraestructuras de red, *middleware* de seguridad y autenticación, así como aspectos organizativos: dotación y gestión de entornos y salas de videoconferencia, normativas y procedimientos de administración de los servicios, etc. Prestaremos especial atención al protocolo SIP, protocolo de señalización propuesto por el IETF como alternativa a la recomendación H.323 de la ITU-T.

3.2. FUNCIONALIDAD PROPORCIONADA POR SIP

El protocolo de señalización SIP (*Session Initiation Protocol*), en términos generales, sirve para establecer, modificar y terminar sesiones multimedia. Pede ser utilizado ya sea para crear una nueva sesión o para invitar nuevos miembros a una ya existente.

3.2.1. ESTABLECIMIENTO, MODIFICACIÓN Y TERMINACIÓN DE SESIÓN

El protocolo SIP es independiente del tipo de sesión multimedia y del mecanismo utilizado para describirla; es igualmente utilizado para videoconferencias, llamadas, compartir recursos o sesiones de juego. En resumen, el protocolo SIP es utilizado para distribuir la descripción de las sesiones entre los participantes potenciales (Fig 3.1). Una vez que la descripción de la sesión es distribuida, el protocolo SIP puede ser usado para negociar y modificar los parámetros así como terminar la sesión.

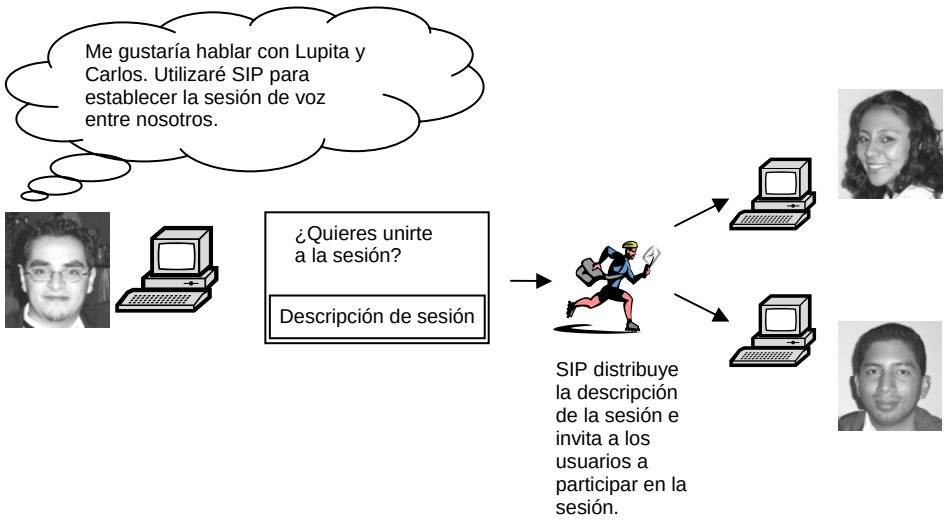


Figura 3.1 Juan Luis invita a Carlos y a Lupita a una sesión de voz

El siguiente ejemplo muestra todas las funciones del protocolo. Juan Luis quiere tener una sesión de audio y video con Lupita y planea usar un CODEC PCM (*Pulse Code Modulation*) para codificar la voz. En este ejemplo, la parte de la distribución de la sesión cosiste en que Juan Luis le envíe a Lupita una descripción de la sesión indicando que el quiere utilizar PCM para codificar la voz, sin embargo, Lupita prefiere utilizar un CODEC GSM (*Global System for Mobile Communications*) porque éste consume un menor ancho de banda, así que Lupita se encarga de persuadir a Juan Luis para que se haga de esa manera. Ambos finalmente acuerdan utilizar el CODEC de audio GSM, pero la sesión no se puede establecer hasta que esta negociación ha concluido.

De repente, en medio de la sesión de audio y video, Lupita decide interrumpir la transmisión del video, así que modifica la sesión de tal manera que sea únicamente de audio. Después de un rato, finalmente Juan Luis decide concluir la conversación y la sesión termina.

De la misma manera como los aparatos de la red telefónica pública informan a la persona a cerca del estado de la llamada por medio de diferentes tonos (tono de marcar, ocupado o sonando), de igual manera el protocolo SIP informa al iniciador de la sesión a cerca del progreso de su llamada por medio de paquetes de información (Fig. 3.2).

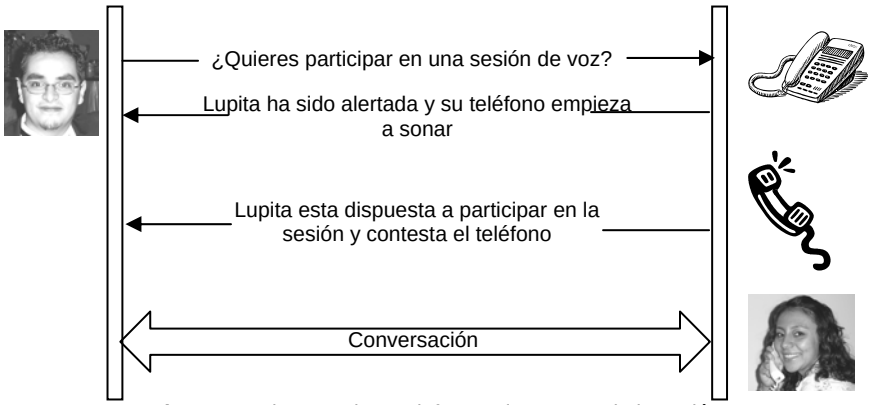


Figura 3.2 El protocolo SIP informa el progreso de la sesión

3.2.2. MOVILIDAD DEL USUARIO.

El protocolo SIP no puede establecer una sesión con un posible participante hasta que éste sea ubicado. Frecuentemente, un único usuario puede ser localizado en diferentes lugares. Por ejemplo, un estudiante que trabaja comúnmente en diferentes computadoras cada día, puede ser contactado mediante diferentes direcciones IP (*Internet Protocol*) dependiendo de que computadora este disponible. Otra persona por ejemplo, quiere establecer una sesión en su estación de trabajo cuando llega a su oficina, en la computadora de su casa por la tarde y en su terminal móvil cuando se encuentra de viaje.

3.2.2.1. SIP URLS

Como ya se ha mencionado, el protocolo SIP provee una cierta movilidad a los usuarios, esta movilidad en el ambiente de SIP es proporcionada por una dirección URL (*Uniform Resource Locators*), mediante la cual los usuarios pueden ser identificados. El formato de la SIP URL es similar a una dirección de e-mail,

generalmente consiste en un nombre de usuario y un nombre de dominio como la que se muestra a continuación; SIP:Lupita.Vazquez@softel.com.

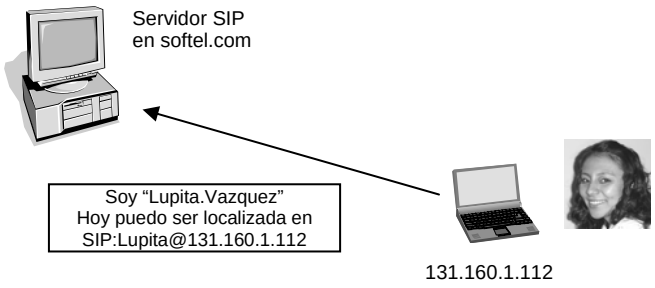
En el ejemplo anterior, si se consulta al servidor SIP que maneja el dominio de softel.com, seguramente encontraremos un usuario cuyo nombre es Lupita.Vazquez.

La URL de Lupita puede ser SIP:Lupita@131.160.1.112, indicando que el *host* cuya dirección IP es 131.160.1.112 esta relacionado con un usuario cuyo nombre de usuario es Lupita.

3.2.2.2. REGISTROS.

Cada usuario debe de registrar su actual posición en un servidor, de tal manera que pueda ser localizado. En este ejemplo, Lupita se encuentra trabajando en una computadora de la escuela, cuya dirección IP es 131.160.1.112, su login es Lupita. Ella registra su posición actual con el servidor de la compañía como se muestra en la figura (Fig 3.3).

Figura 3.3 Lupita registra en el servidor su posición actual.



Ahora, Carlos quiere establecer una comunicación con ella, el tiene su dirección SIP pública (SIP:Lupita.Vazquez@softel.com.), por lo que cuando el servidor de softel.com es contactado para preguntar por Lupita.Vazquez, él sabe donde puede ser contactada y la conexión es establecida.

En esta situación, el protocolo SIP provee dos formas de operación: redireccionada o mediante un proxy. Mediante el proxi, el servidor contacta a Lupita en 131.160.1.112 y le entrega la descripción de la sesión de Carlos (Fig 3.4).

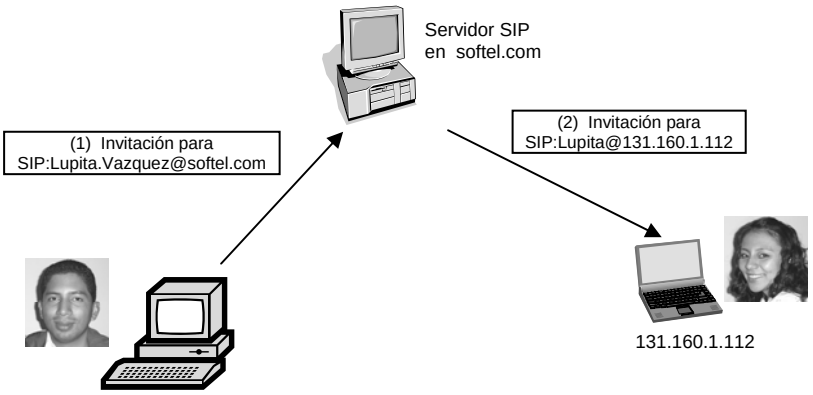


Figura 3.4 Servidor Proxy de SIP

En el modo redireccionado, el servidor de indica a Carlos que utilice la dirección Lupita@131.160.1.112, en lugar de la dirección pública que estaba utilizando.

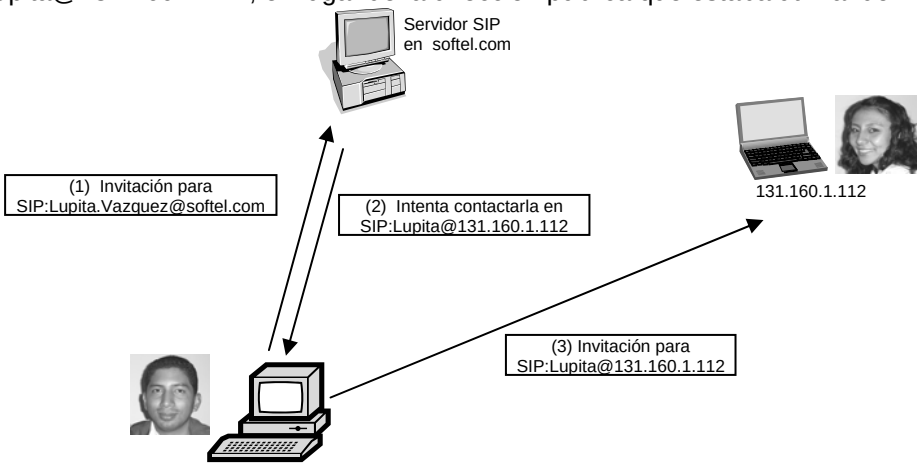


Figura 3.5 Servidor SIP de redireccionamiento

3.3. ARQUITECTURA DEL PROTOCOLO SIP

Las arquitecturas definidas en torno a los protocolos H.323 y SIP son sustancialmente distintas; sin embargo, al comparar la evolución de ambos estándares durante los últimos años, se extrae la conclusión de que, a medida que se definen nuevas ampliaciones a SIP y aparecen nuevas versiones de H.323, cada vez se diferencian menos en cuanto a las funciones y posibilidades que ofrecen.

Tanto la recomendación H.323 como el protocolo SIP definen mecanismos de señalización para establecer y terminar llamadas, así como otras funciones de control de conferencia, negociación de capacidades y servicios adicionales sobre redes de conmutación de paquetes. SIP se ha diseñado con posterioridad con la pretensión de que, desde la perspectiva de los estándares y prácticas habituales en Internet, presente las siguientes ventajas frente a H.323:

- Implementación más fácil de realizar y depurar
- Mayor flexibilidad para incorporar nuevas funciones
- Mayor integración con otras aplicaciones y servicios Internet

Aunque SIP se creó como un protocolo de inicio de sesiones, ha evolucionado, mediante la definición de nuevas funciones y servicios en forma de módulos complementarios basados en un núcleo de funciones básicas flexibles y ampliables, hasta constituirse en el protocolo de señalización y control propuesto por el IETF como base para los servicios de telefonía y comunicación multimedia en general en Internet, así como el protocolo de señalización de la red telefónica de tercera generación y la base de algunos de los sistemas de mensajería instantánea más extendidos.

SIP es un protocolo de señalización cliente-servidor de nivel de aplicación válido para redes unicast y multicast. Generalmente, los mensajes SIP constan de un conjunto de cabeceras y un cuerpo que contiene descripciones de sesiones multimedia, siendo SDP el formato utilizado en la actualidad.

Puesto que el formato de los mensajes SIP es textual, basado en HTTP y SMTP, y sigue principios similares a los de HTTP, es posible desarrollar servicios SIP mediante los procedimientos extendidos en la Web. SIP cumple las siguientes funciones: establecimiento,

modificación y finalización de sesiones multimedia, registro y localización de participantes, gestión del conjunto de participantes y de los componentes del sistema, así como descripción de las sesiones y negociación de capacidades.

Los dos tipos de elementos presentes en la arquitectura SIP son los *agentes de usuario (UA)* y los *servidore*s.

3.3.1. AGENTE DE USUARIO SIP

El Agente de Usuario (*User Agent*) es el software de SIP en el punto terminal o estación terminal. Funciona como un cliente cuando hace las peticiones de inicio de sesión (UAC), y también actúa como un servidor cuando responde a las peticiones de sesión (UAS). Por tanto, la arquitectura básica es de naturaleza cliente/servidor. El Agente de Usuario (UA) es "inteligente", en el sentido que almacena y administra el estado de la llamada. El UA establece las llamadas usando una dirección parecida a las de correo electrónico, o un número telefónico. Por ejemplo: SIP:usuario@servidor.universidad.edu. Esto hace que los URL de SIP sean fáciles de asociar con la dirección de correo electrónico del usuario. Los siguientes elementos dan funcionalidades y niveles de administración extra al esquema SIP.

3.3.2. SERVIDORES SIP

Las funciones básicas de los servidores SIP son la localización de usuarios y la resolución de nombres. Puesto que generalmente los agentes de usuario clientes no conocen la dirección IP del destinatario de una llamada, sino su nombre de usuario o un número de teléfono al que habrá que acceder a través de un *gateway*, necesitan enviar en primer lugar un mensaje de invitación al servidor correspondiente al nombre o número para que localice al destinatario. El servidor puede conocer la dirección del destinatario o recurrir a otros servidores para continuar la búsqueda. Cuando las llamadas se redirigen, la ruta seguida se registra en los mensajes SIP, de modo que a la hora de generar respuestas se pueda conocer el camino de retorno hasta el origen del mensaje inicial.

Los servidores SIP actúan generalmente como varios tipos de servidores de forma simultánea. Gracias a una infraestructura de servidores SIP, es posible gestionar las llamadas de forma distribuida entre equipos personales, equipos de proveedores de servicios y pasarelas corporativas, con la consiguiente flexibilidad y control por parte del usuario, que puede mantener la privacidad de sus datos personales en todo momento. Asimismo, un mensaje SIP puede pasar por un número indeterminado de servidores desde que un agente de usuario cliente lo envía hasta que llega al agente de usuario servidor destinatario.

Los servidores SIP pueden ser de tres tipos no excluyentes:

SERVIDOR DE REDIRECCIÓN

A diferencia de los *proxies*, no inician transacciones, sino que, cuando reciben solicitudes desde un agente de usuario cliente, remiten al mismo agente un mensaje indicando el o los servidores con los que debe ponerse en contacto, en un procedimiento similar al de búsqueda iterativa del sistema DNS. Asimismo, a diferencia de los agentes de usuario servidores, no aceptan llamadas.

Normalmente, los servidores de redirección gestionan mayor número de mensajes que los *proxies*, pero con menores necesidades de procesamiento. Nótese que, puesto que en sesiones controladas por SIP la redirección se realiza mediante

mensajes SIP, las respuestas se pueden generar con flexibilidad y adecuación a servicios de conferencia multimedia, modificándose en función de parámetros tales como la hora del día, el origen o urgencia de la llamada, o cualquier otro criterio específico aplicado por el servidor SIP.

SERVIDOR DE REGISTRO

El uso más común de estos servidores es registrar un dispositivo después de su arranque, de modo que cuando lleguen invitaciones destinadas a él, los servidores SIP puedan proporcionar su dirección. Se contempla la existencia de un tiempo máximo de validez de cada registro, definible por el servidor, tras el cual se debe renovar el registro. Asimismo, existen mecanismos para cancelar todos los registros contenidos en un servidor.

SERVIDOR PROXY

Otro tipo de servidor intermedio es el Servidor Proxy SIP. Los Servidores Proxy reenvían peticiones desde el Agente de Usuario hacia el siguiente Servidor SIP, y retienen la información por cuestiones de contabilidad o facturación. Adicionalmente, el Servidor Proxy SIP puede operar en forma constante (como un circuito) o dependiente de la conexión (vía TCP). El Servidor constante SIP puede dirigir las llamadas entrantes hacia diversas extensiones que están activas a la vez y la primera en responder tomará la llamada. Esta capacidad significa que se puede especificar el teléfono SIP en el escritorio, el teléfono móvil SIP y la aplicación de videoconferencia en casa de tipo SIP y todos esos aparatos "sonarían" cuando llegue una llamada que está tratando de localizar al usuario, de tal forma que al contestar en cualquiera de esos medios se inicia la conversación y los otros dispositivos dejan de sonar. Los Servidores Proxy SIP pueden usar varios métodos para intentar resolver la dirección destino solicitada, incluyendo búsquedas en el DNS, en bases de datos o relevando la labor hacia el siguiente Servidor Proxy.

Los proxys son, al igual que los proxys HTTP, particularmente útiles como representantes de salida/entrada de y a redes corporativas, proporcionando servicios de búsqueda de direcciones, control de cortafuegos y gestión de normas de administración corporativas. Asimismo, pueden cumplir funciones de control de salida a pasarelas para redes telefónicas tradicionales.

Aunque una llamada básica se puede realizar con SIP sin que intervengan servidores, las funciones avanzadas del protocolo no se pueden llevar a cabo sin su participación. La interacción entre UAC, UAS y servidores se realiza mediante el intercambio de mensajes SIP.

3.4. ANUNCIO DE SESIONES: SAP

La especificación actual del protocolo SAP establece las reglas que los directorios de sesiones multicast deben seguir en la transmisión periódica de anuncios de sesiones, tales como el límite de ancho de banda, el algoritmo a seguir para el cálculo del intervalo de transmisión, las direcciones y el puerto de destino de los anuncios, o las condiciones bajo las cuales se considera que las sesiones anunciadas dejan de estar activas. Puesto que SAP es un protocolo diseñado para el anuncio de sesiones predefinidas, a diferencia de SIP, no contempla ningún proceso de negociación, sino que los anuncios establecen citas para la celebración de sesiones con características fijadas por el anunciante.

Los anuncios SAP contienen la descripción de sesiones junto con una cabecera de autenticación. Aunque se admite el uso de formatos distintos a SDP para describir sesiones,

se desaconseja en aras de la mayor interoperabilidad posible, mientras que es obligatorio que todo agente SAP manipule correctamente descripciones en formato SDP. Dado que SAP es un protocolo con muy bajo consumo de ancho de banda –por omisión se asume cuatro *kilobits* por segundo en cada grupo de anuncios–, el periodo de transmisión de anuncios propuesto por el estándar puede llevar a que cada sesión se anuncie a intervalos del orden de decenas de minutos. Por ello, como mecanismo para que sea posible determinar rápidamente la existencia de sesiones, se recomienda el despliegue de proxys caché que mantengan una lista actualizada de todas las sesiones anunciadas.

3.5. DESCRIPCIÓN DE SESIONES Y NEGOCIACIÓN DE CAPACIDADES: SDP

SIP claramente distingue entre el establecimiento y la descripción de la sesión. Como parte del establecimiento de la sesión, el protocolo SIP localiza a los usuarios como ya se mencionó, pero es transparente respecto a lo que los usuarios pueden hacer una vez que la sesión se he establecido. *El protocolo SIP solo provee conectividad, no define ni el tipo de sesión ni como esta debe ser descrita,* para esto, se tiene el protocolo SDP (*Session Description Protocol*), el cual especifica como debe ser codificada la información necesaria para describir una sesión. Este protocolo no incluye ningún mecanismo de transporte ni ningún tipo de parámetro de negociación. La descripción de SDP es simplemente un pedazo de información que el sistema puede usar para incluirse en una sesión multimedia. Esto puede ser por ejemplo: la dirección IP, el número de los puertos y los horarios precisos en los que la sesión está activa.

Una buena analogía para entender esto, podría ser la información que nos proporcionan las guías de televisión, la descripción de la sesión podría ser la siguiente: “Sintonizar el canal 2 a las 9:00 pm diariamente para ver las noticias”. Esta información nos provee información de dónde recibir la información (canal 2), cuándo la señal va a ser transmitida (diariamente a las 9:00 pm) y la información que contiene la sesión (noticias).

El protocolo SIP es capaz de trasladar esta información (SDP) dentro de sus propios mensajes.

3.5.1. SINTAXIS DEL PROTOCOLO SDP

Es importante revisar la sintaxis del protocolo SDP, ya que aparece en muchos mensajes de SIP. Las descripciones de SDP están basadas en texto. Una descripción de sesión SDP consiste en un conjunto de líneas de texto con la siguiente forma:

Type = value

Una descripción SDP contiene dos tipos de información: una a nivel de sesión y otra que se aplica a nivel de los medios de comunicación, ya sea voz, video, ambas, etc. La información de nivel de sesión se aplica durante el transcurso de ésta, y puede ser, por ejemplo el origen o el nombre de dicha sesión. La información a nivel de los medios se refiere por ejemplo a el codec utilizado para enviar los paquetes de voz o el número de puerto donde van a ser enviados los paquetes de video.

La descripción de una sesión SDP empieza con la información de nivel de sesión y si es que contiene información de nivel de medios, ésta viene después. La sección de nivel de sesión, siempre empieza con v=0, donde *v* es el tipo y 0 es el valor. Esta línea indica que la versión del protocolo es 0 (SDP versión 0).
Seguido de esta línea se encuentra la información de toda la sesión, hasta la primera línea de información de nivel de medios, si es que la descripción contiene, de lo contrario, hasta el final de dicha descripción.

La sección de los datos de nivel de medios se distingue con una línea que empieza con la letra *m*. Las líneas siguientes proveen información de esa aplicación en particular. El siguiente es un ejemplo de una descripción de sesión SDP:

```
v=0
o=Juan Luis  2890844526 2890842807 IN IP4 131.160.1.112
s=Fiest
i=Junta para organizar fiesta
u=http://www.cs.comunidad.com/fiesta
e=jluis@unam.edu
c=IN  IP4  131.160.1.112
t=2873397496  2873404696
a=recvonly
m=audio  49170  RTP/AVP  0
a=rtpmap:0  PCMU / 8000
m=video  51372 RTP/AVP  31
a=rtpmap:31 H261/90000
m=video  53000 RTP/AVP  32
a=rtpmap:32 MPV / 90000
```

En este ejemplo la sección de nivel de sesión esta compuesta de las primeras nueve líneas, desde ***v=0*** hasta ***a=recvonly***, y tiene tres secciones de nivel de medios: una de audio y dos de video. La línea ***o*** indica quien fue el iniciador de la sesión (en este caso, Juan Luis) y la dirección IP que le pertenece. La línea ***s*** contiene el nombre de la sesión y la línea ***i*** contiene información general a cerca de la sesión. La línea ***u*** provee una URL de donde mas información acerca del propósito de la sesión puede ser capturada. La línea ***e*** contiene el e-mail de la persona que inicia la sesión. La dirección de *multicast* donde la sesión puede ser recibida se indica en la línea ***c*** y la línea ***t*** indica cuando la sesión esta activa. La última línea de la sección de nivel de sesión, la línea ***a,*** indica que no es una sesión interactiva, es una sesión solo para recibir información.

El formato de la línea ***m*** es especialmente importante. Una línea ***m*** empieza con el tipo de medio del que se trate. En el ejemplo anterior, el primer tipo de información era audio y los siguientes dos video.

```
m=<tipo de información><número de puerto><protocolo de transporte><formato de información>
```

El número de puerto indica donde será recibida la información. El campo de protocolo de transporte usualmente toma el valor de RTP/AVP, pero puede tomar otro valor si es utilizado un protocolo diferente del RTP. RTP/AVP, se refiere al perfil de audio/video para RTP; en este ejemplo, el audio y video codificados, son transportados usando RTP sobre UDP.

El formato de la información depende del tipo de información de medios que se transporte. Para audio, podría ser el CODEC utilizado. En este ejemplo, el valor de 0 indica que el audio es codificado en un solo canal utilizando PCM con la ley μ y muestreado a 8 kHz.

La línea ***a=rtpmap*** transporta información como la velocidad del reloj o el número de canales utilizados. En la segunda línea ***m*** del ejemplo, el número 31 se refiere al H.261 y éste utiliza una velocidad de reloj de 90 kHz.

La siguiente tabla contiene todos los tipos definidos por SDP y su significado correspondiente.

v	Versión del Protocolo
b	Información del ancho de banda
o	Propietario de la sesión e identificador de la sesión
z	Ajustes de la zona horaria
s	Nombre de la sesión
k	Llave de encripción
i	Información sobre la sesión
a	Líneas de atributos
u	URL que contiene una descripción de la sesión
t	Tiempo cuando la sesión esta activa
e	Dirección de mail para obtener información de la sesión
r	Tiempo en el que la sesión será repetida
p	Número telefónico para obtener información de la sesión
m	Línea de información
c	Información de conexión
i	Información sobre la línea de información

Tabla 3.1 Tipos de SDP

3.5.2. SDP EXTENDIDO

Las líneas de atributos de la información, las *a* líneas, proveen una extensión de SDP. Cuando una aplicación necesita una característica que no contiene el SDP, ésta puede ser añadida en una línea *a.* Por ejemplo, si el creador de una sesión quiere que los receptores toquen el audio a cierto volumen, éste puede definir un nuevo atributo a la información e incluirlo al final de la sección de nivel de medios.

```
m=audio 49170 RTP/AVP 0
a=volume:8
```

Las aplicaciones que entienden esta nueva línea *a* tocarán el audio a un volumen de 8. Cuando una aplicación encuentra una línea *a* que no entiende, simplemente la ignora y procede como si no la hubiera encontrado. La aplicación que no entienda las líneas *a*, puede seguir recibiendo la información adecuadamente aunque no tocará el audio con el volumen adecuado.

3.6. OPERACIÓN DEL PROTOCOLO SIP

En esta parte se describirá el protocolo SIP más detalladamente, como logra su funcionalidad: esto es, que mensajes son intercambiados entre las diferentes entidades SIP y cuáles son los formatos de estos mensajes.

3.6.1. TRANSACCIONES CLIENTE SERVIDOR

El protocolo SIP esta basado en el protocolo Web *Hypertext Transfer Protocol* (HTTP) y como él, el protocolo SIP es un protocolo de petición/respuesta. Para entender este mecanismo se tendrán que examinar las siguientes definiciones de cliente y servidor.
Un *cliente* es una entidad de SIP que genera peticiones. Un *servidor* es una entidad de SIP que recibe peticiones y las responde.

Esta terminología es propia de HTTP, donde un buscador Web contiene un cliente HTTP. Cuando se escribe una dirección en un buscador Web como la siguiente http://www.hotmail.com, en realidad se está enviando una petición a un servidor Web particular. El servidor Web regresa una respuesta con la información solicitada, esto es, la página Web requerida.

El protocolo SIP utiliza el mismo procedimiento. Siguiendo la misma terminología, cuando dos agentes de usuario intercambian mensajes de SIP, el agente de usuario (UA) que envía peticiones es el agente de usuario cliente (UAC) y el UA que responde a esas peticiones, es el agente de usuario servidor (UAS). Una petición SIP junto con la respuesta que produce es conocida como una *transacción* SIP.

3.6.1.1. RESPUESTAS DE SIP

Sobre la recepción de una petición, un servidor emite una o varias respuestas. Cada respuesta tiene un código que indica el estado de la transacción. Los códigos de estado son números enteros en un rango de 100 a 699 y están agrupados en clases como se muestra en la tabla siguiente.

Tabla 3.2 Tipos de respuestas de SIP

Rango	Tipo de respuesta
100-199	Informativa
200-299	Exitosa
300-399	Redirección
400-499	Error del cliente
500-599	Error del servidor
600-699	Falla global

Una respuesta con un código de estado de 100 a 199 es considerada como provisional. Respuestas entre 200 y 699 son respuestas finales. Una transacción SIP entre un cliente y un servidor esta formada generalmente por una petición del cliente, una o más respuestas provisionales y una respuesta final.

Junto con el código de estado, las respuestas de SIP llevan consigo información que permite a una persona entender este código de estado. Por ejemplo, el código 180 significa que el usuario invitado a un a sesión ha sido alertado, por lo tanto, la frase que acompañe al código de estado puede ser “Ringing” (sonando). Esta frase

puede ser escrita en cualquier lenguaje puesto que va a ser leído por una persona y no importa que la computadora no lo interprete. Desde ahora, se citarán las respuestas usando el código de estado seguido de la frase que lo interpreta, por ejemplo, “180 Ringing”. La tabla 3.3 contiene todos los códigos de estado actualmente definidos con su frase asociada.

100	Trying	413	Request entity too large
180	Ringing	414	Request-URI too large
181	Call is being forwarded	415	Unsupported media type
182	Queued	420	Bad extensión
183	Session progress	480	Temporarily not available
200	OK	481	Call leg/transaction does not exist
202	Accepted	482	Loop detected
300	Multiple choices	483	Too many hops
301	Moved permanently	484	Address incomplete
302	Moved temporarily	485	Ambiguous
305	Use proxy	486	Busy here
380	Alternative service	487	Request cancelled
400	Bad request	488	Not acceptable here
401	Unauthorized	500	Internal server error
402	Payment required	501	Not implemented
403	Forbidden	502	Bad gateway
404	Not found	503	Service unavailable
405	Method not allowed	504	Gateway time-out
406	Not acceptable	505	SIP version not supported
407	Proxy authentication required	600	Busy everywhere
408	Request time-out	603	Decline
409	Conflict	604	Does not exist anywhere
410	Gone	606	Not acceptable
411	Length required		

Tabla 3.3 Códigos de respuesta de SIP

3.6.1.2. PETICIONES DE SIP

La especificación SIP define seis tipos de peticiones cada una con diferentes propósitos. Cada petición de SIP contiene un campo llamado *método,* el cual denota su propósito. La lista muestra los seis métodos existentes.

- *INVITE*
- *ACK*
- *OPTIONS*
- *BYE*
- *CANCEL*
- *REGISTER*

Cada mensaje de SIP (peticiones o respuestas), contiene una carga útil (payload) que usualmente consiste en la descripción de la sesión (SDP).

3.6.1.2.1. *INVITE*

Las peticiones del método INVITE, como su nombre lo indica, invitan a los usuarios a participar en una sesión. El cuerpo de las peticiones del INVITE contienen la descripción de la sesión. Por ejemplo cuando Juan Luis llama a Lupita, su UA envía un INVITE con una descripción de la sesión al UA de Lupita

El UA de Lupita recibe un mensaje de INVITE con la siguiente descripción de sesión:

```
v=0
o=Juan Luis  2890844526 2890842807 IN IP4 131.160.1.112
s=Quiero saludarte
c=IN  IP4  131.160.1.112
t=0 0
m=audio  49170  RTP/AVP 0
```

El INVITE recibido por Lupita indica que Juan Luis la está invitando a unirse a una sesión de audio. Por la descripción de la sesión que lleva el mensaje de INVITE el UA de Lupita sabe que Juan Luis quiere recibir paquetes de RTP (*Real Time Transport Protocol*) que contengan su voz, a la dirección 131.160.1.112 por el puerto 49170 de UDP (*User Datagram Protocol*); también sabe que Juan Luis puede recibir voz codificada en PCM (*Pulse Code Modulation*). (La información RTP/AVP 0, en la línea *m,* indica PCM).

El UA de Lupita empieza a alertarla y regresa un “180 Ringing” al UA de Juan Luis. Cuando Lupita finalmente acepta la llamada, su UA regresa un mensaje de “200 OK” con una descripción de sesión en él.

```
v=0
o=Lupita  281234526 281234287 IN IP4 138.85.27.10
s=Quiero saludarte
c=IN  IP4  138.85.27.10
t=0 0
m=audio  20000  RTP/AVP  0
```

En este punto, Lupita aceptó la llamada e informa a Juan Luis que ella recibirá paquetes RTP en la dirección IP 138.85.27.10 por el puerto UDP 20000 (Fig. 3.6).

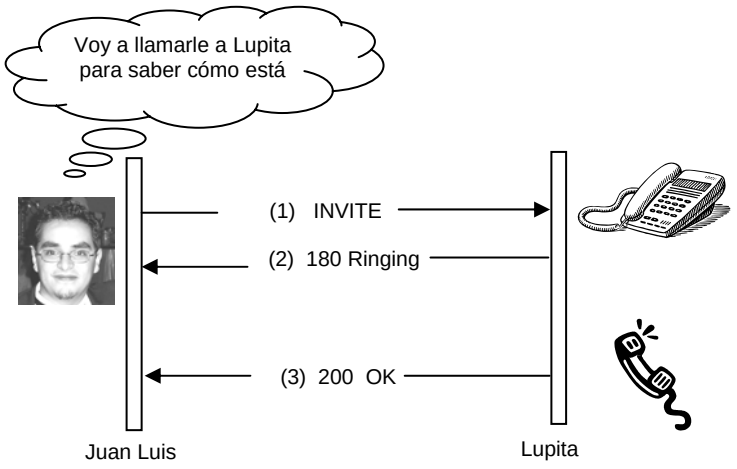


Figura 3.6 Lupita envía una respuesta final (200 OK) a el INVITE que recibió.

3.6.1.2.2. *ACK*

La petición de ACK es usada para confirmar la recepción de una respuesta final a un INVITE. Así, un cliente que origina un INVITE, emite un ACK cuando recibe una respuesta final al INVITE.

El método INVITE es el único que emite una confirmación después de haber recibido una respuesta final, a diferencia de los demás métodos que utilizan un modelo MÉTODO-respuesta final, a este procedimiento se le conoce como *handshake* de tres vías (Fig. 3.7).

El *handshake* de tres vías, hace posible la implementación de los *proxies* de bifurcación. Cuando se envía una petición a través de uno de estos *proxies*, el UA que la envió recibe varias respuestas de diferentes servidores por lo que mandar un ACK a cada destino que respondió es básico para asegurar el funcionamiento de SIP sobre protocolos poco confiables como lo es UDP.

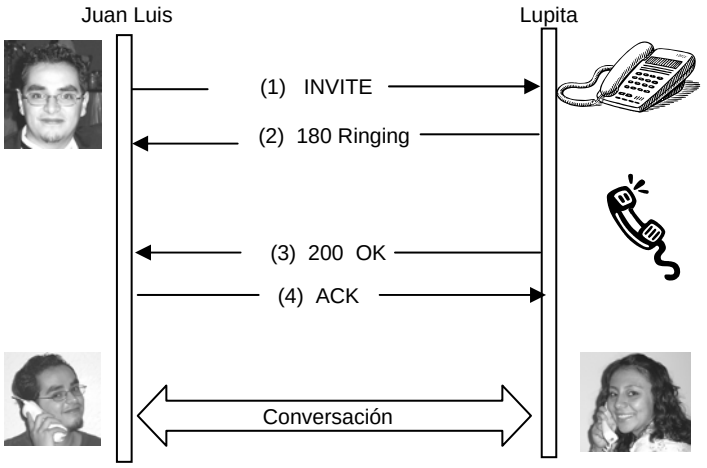


Figura 3.7 Handshake de tres vías.
INVITE-200 OK-ACK

Algunas características hacen que el método INVITE sea diferente de los demás métodos. Cuando un cliente hace una petición diferente de un INVITE, espera una respuesta rápida por parte del servidor. Sin embargo, la respuesta a un INVITE puede tomar bastante tiempo. Cuando Juan Luis llama a Lupita,

ella tiene que caminar a donde se encuentra el teléfono y oprimir algunos botones, por lo que la respuesta de “200 OK” puede estar más o menos retrasada, depende el tiempo que le tarde a Lupita contestar, por lo que enviando un ACK del cliente al servidor cuando la respuesta es recibida, el servidor puede saber que el cliente está todavía allí (si no se cansó de esperar) y que la sesión fue establecida con éxito (Fig. 3.8).

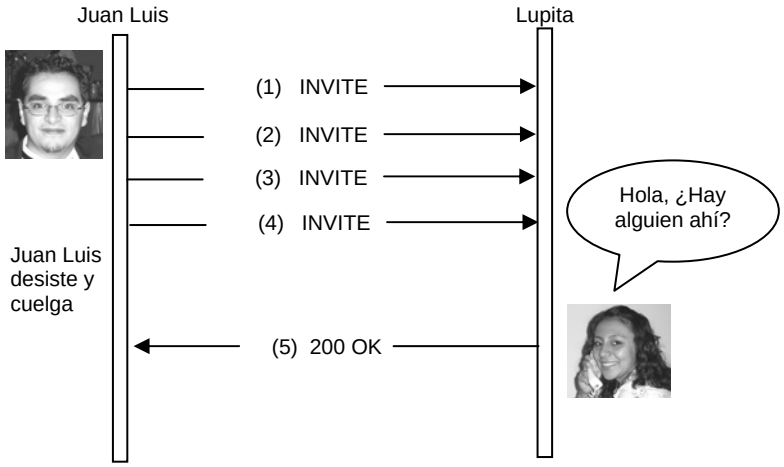


Figura 3.8 Esta situación puede evitarse con el handshake de tres vías

3.6.1.2.3. *CANCEL*

La petición de CANCEL, como su nombre lo indica se utiliza para cancelar transacciones pendientes. Si un servidor SIP recibió un INVITE pero no ha enviado una respuesta final, podrá detener el proceso del INVITE si recibe una petición de CANCEL. Si aun recibiendo el CANCEL, el servidor envía una respuesta final, entonces la transacción se llevará a cabo y la petición de CANCEL no tendrá ningún efecto.

En la figura 3.9, Carlos llama a Lupita, su teléfono SIP empieza a sonar y continúa sonando durante algún tiempo sin que nadie lo conteste, así que Carlos decide colgar; al hacer esto, envía un CANCEL al INVITE que previamente había enviado. Como consecuencia del CANCEL, el teléfono de Lupita deja de sonar. El servidor, contesta un 200 OK al CANCEL indicando que fue procesado satisfactoriamente.
Es importante remarcar que después de que el servidor ha respondido al CANCEL, de todas maneras debe responder al INVITE, él manda un “487 Transaction Cancelled” y el cliente concluye el handshake de tres vías enviando un ACK (INVITE-487 Transaction Cancelled-ACK). El handshake de tres vías del INVITE es siempre concluido, incluso cuando la transacción es cancelada.

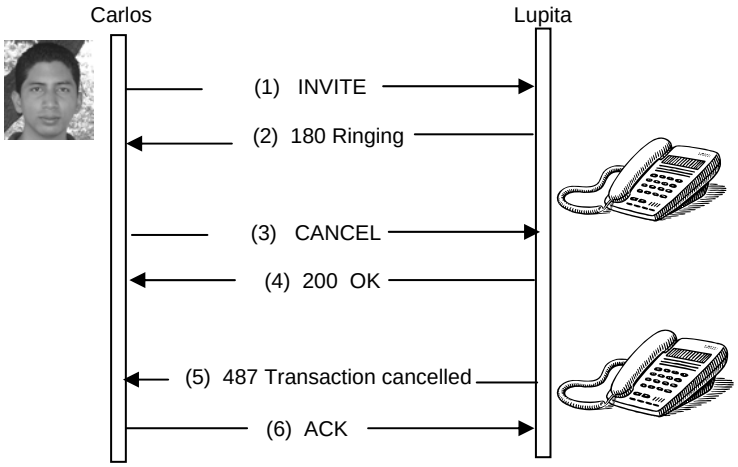


Figura 3.9 Carlos cancela su invitación.

La petición de CANCEL es muy útil cuando en el camino se encuentran *proxies* de bifurcación (éstos al recibir un INVITE lo envían a los diferentes lugares donde la persona puede ser localizada).

Cuando este tipo de *proxy* se encuentra haciendo una búsqueda paralela, intenta contactar a la persona en diferentes lugares al mismo tiempo, por ejemplo, un *proxy* de bifurcación conoce tres posibles lugares donde Carlos puede ser contactado:

- SIP:Carlos@131.160.1.112
- SIP:Carlos.Villanueva@softel.com
- SIP:Carlos@unam.com.

Cuando este *proxy* recibe un INVITE de Lupita a Carlos, intentará localizarlo en estas tres direcciones al mismo tiempo. Carlos, que se encuentra actualmente en 131.160.1.112 contesta la llamada. El *proxy* recibe un 200 OK de SIP:Carlos@131.160.1.112 y lo reenvía al UA de Lupita; como la sesión entre ellos logra establecerse, el *proxy* de bifurcación quiere detener las otras búsquedas iniciadas, así que envía dos *CANCEL*s, una para cada dirección (Fig. 3.10).

Es importante recordar, que un *CANCEL* no afecta a la transacción una vez que la respuesta final ha sido enviada, por lo que en el ejemplo anterior, incluso si el *proxy* enviara un CANCEL a SIP:Carlos@131.160.1.112, la sesión entre Carlos y Lupita se mantendría activa. Un CANCEL no puede terminar una transacción en curso y es ignorado por las transacciones completas.

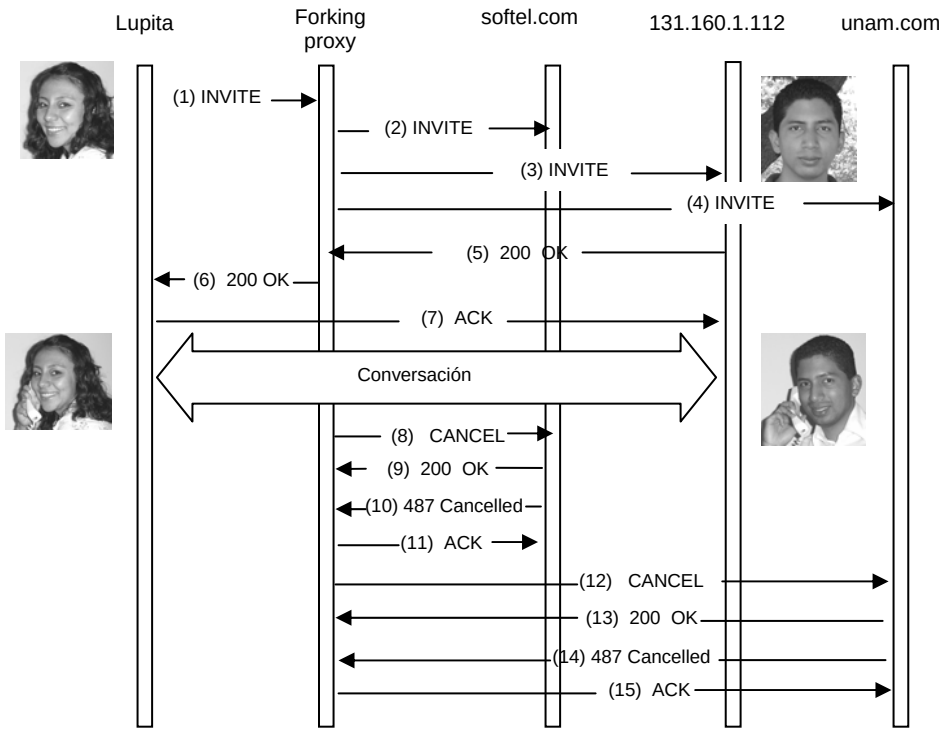


Figura 3.10 JEl servidor Proxy cancela las transacciones de INVITE

3.6.1.2.4. *BYE*

La petición de BYE es utilizada para abandonar la sesión. En una sesión con solo dos integrantes, el hecho de que alguno de los dos abandone la sesión indica que ésta ha sido terminada (Fig. 3.11). Por ejemplo, cuando Carlos le envía un BYE a Lupita, su sesión es automáticamente terminada, sin embargo, en conversaciones entre mas personas, un mensaje de BYE de parte de uno de los participantes solo indica que únicamente ese participante abandona la sesión.

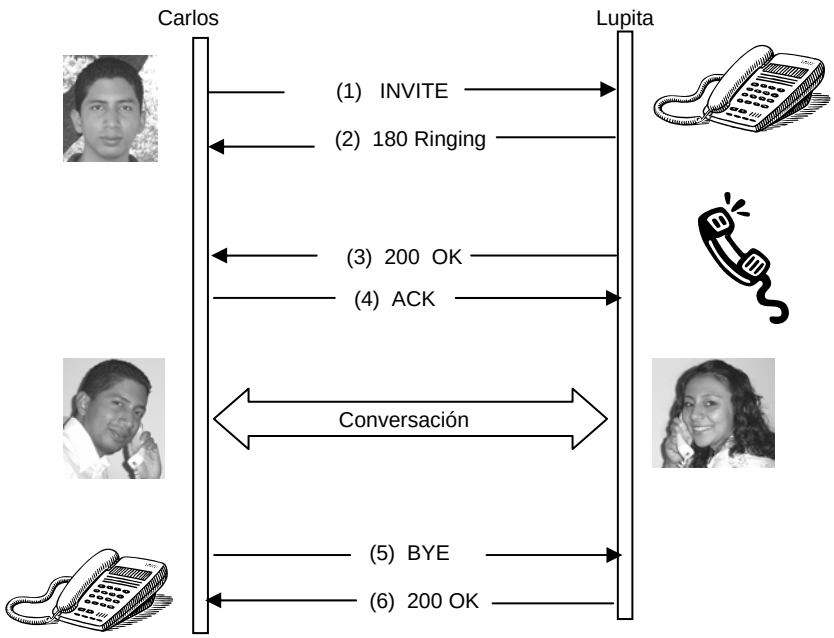


Figura 3.11 Carlos envía un BYE cuando cuelga

3.6.1.2.5. *REGISTER*

Los usuarios envían peticiones de REGISTER para informar a un servidor (en este caso el registrar) acerca de su localización actual. Carlos puede enviar un mensaje de REGISTER al registrar de softel.com informando que todas las peticiones que lleguen para SIP:Carlos.Villanueva@softel.com deben ser redireccionadas a SIP:Carlos@131.160.1.112 (Fig. 3.12).

Los mensajes de REGISTER además contienen el tiempo en el que el registro permanece vigente. Por ejemplo, Carlos puede registrar su actual ubicación hasta las 4:00 pm porque sabe que a esa hora se irá de la oficina. Un usuario puede registrarse en diferentes lugares al mismo tiempo, indicándole al servidor que debe buscarlo en cualquiera de esas direcciones hasta ser contactado.

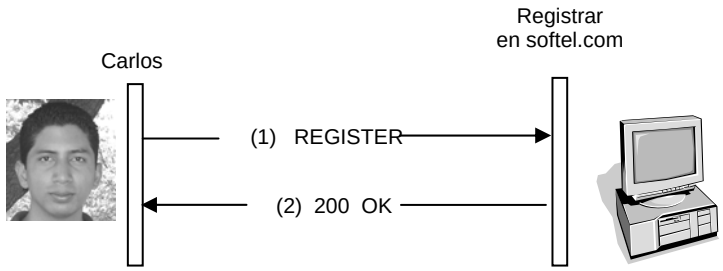


Figura 3.12 Carlos registra su posición actual en el Registrar de softel.com

3.6.1.2.6. *OPTIONS*

Las peticiones de OPTIONS le preguntan al servidor a cerca de su capacidades, incluyendo cuáles métodos y cuáles protocolos de descripción de sesión soporta. Un servidor SIP puede contestar a un OPTIONS que soporta SDP como protocolo de descripción de sesión y cinco métodos: INVITE, ACK,

CANCEL, BYE y OPTIONS. Se puede deducir que este servidor no es un registrar, ya que no soporta el método REGISTER.

El método OPTIONS además, regresa información especificando que tipos de codificación para los cuerpos de mensaje soporta el servidor.

3.7. TIPOS DE SERVIDORES PROXY

Los servidores Proxy pueden ser clasificados de acuerdo a la cantidad de información de estados que pueden almacenar durante una sesión. Los servidores pueden mantener el *estado de las llamadas* a dos niveles de detalle o no mantener estado alguno. Al igual que en la red telefónica convencional, pueden mantener el estado completo de cada llamada; sin embargo, este comportamiento es opcional, puesto que limitaría la escalabilidad de los sistemas SIP. El modo de funcionamiento recomendado y más frecuente es que los servidores mantengan únicamente el estado de cada transacción por separado. Las transacciones SIP se pueden definir como *el conjunto de peticiones y respuestas intercambiadas desde que un agente de usuario cliente envía una petición hasta que recibe una respuesta definitiva originada en el agente de usuario servidor que recibió la petición inicia*l. Por este motivo, si los servidores sólo mantienen el estado de cada transacción por separado, no tienen conocimiento de las llamadas existentes en un cierto instante. Por tanto, los servidores no tienen que mantener una máquina de estados para cada llamada, constituyendo así un sistema altamente escalable. Asimismo, el comportamiento de los servidores SIP en cuanto al mantenimiento de estados se puede modificar de forma dinámica en función de las circunstancias.

3.7.1. PROXY CON MEMORIA COMPLETA DE LOS ESTADOS DE LA LLAMADA

Este tipo de servidores Proxy, necesitan estar informados de todas las transacciones de SIP que ocurran durante la sesión y por lo tanto, están siempre colocados en el camino que toman los mensajes al viajar entre usuarios finales. Estos servidores almacenan la información desde el momento en que la sesión se establece hasta el momento en que se termina.

Por ejemplo, un servidor que implemente un servicio relacionado a la llamada, como el envío de un mail con la información de la duración de ésta, puede ser un servidor Proxy de este tipo, ya que para saber este dato, tiene que estar colocado entre los usuarios, para estar enterado del momento en que se envió el INVITE que inició la llamada, así como del BYE que la termina.

3.7.2. PROXY CON MEMORIA DE ESTADOS POR TRANSACCIÓN

Para estos servidores, la transacción es su único propósito. Este tipo de servidores almacenan información de una transacción dada hasta que esta concluye, además, no necesitan estar en el camino que toman los mensajes de SIP para futuras transacciones.

Los servidores de bifurcación son un buen ejemplo de este tipo de servidores proxy (Fig. 3.13). Ellos mandan INVITEs a diferentes lugares y tienen que almacenar información acerca de la transacción INVITE para saber si alguno de los agentes de usuario (UA) a los que se le envió la petición ha regresado un respuesta final. Sin embargo, una vez que el usuario es contactado en un lugar particular, el servidor Proxy no tiene que permanecer contactado.

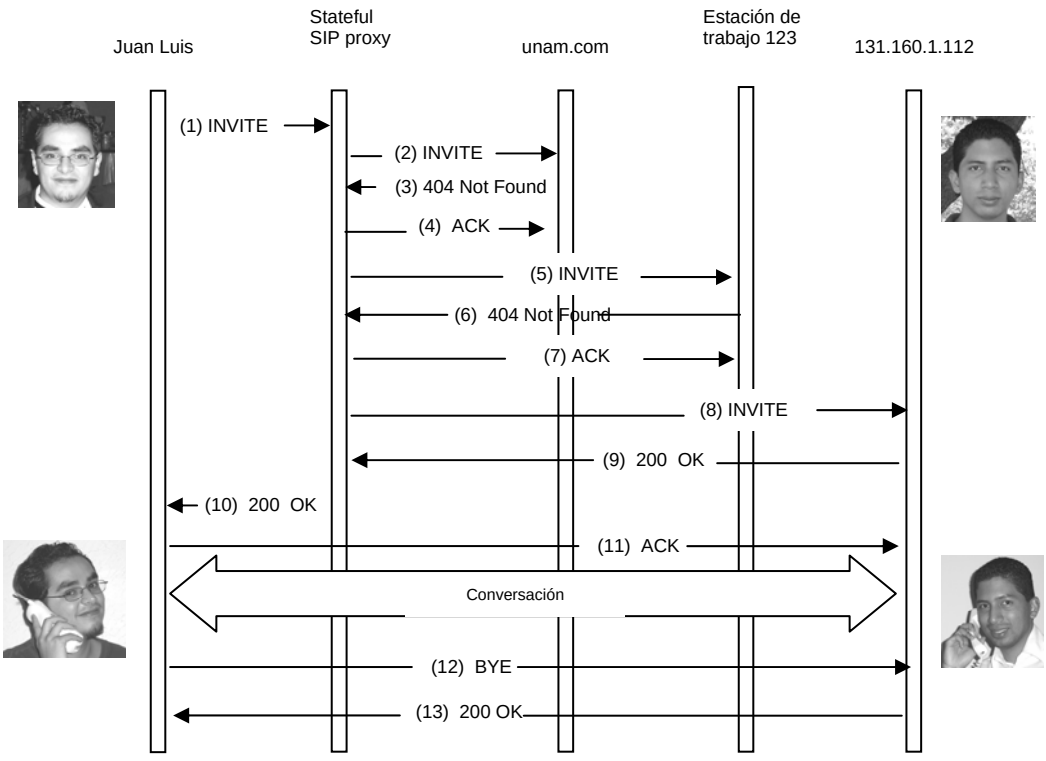


Figura 3.13 Stateful forking proxy

En la figura del ejemplo anterior (Fig. 3.13) podemos ver que el Proxy genera ACKs, sin embargo estos ACKs generados por servidores Proxy solo pueden responder a situaciones no exitosas, aquellas que tengan un código mayor de 299. Respuestas exitosas (códigos de estado entre 200 y 299) son siempre confirmados con un ACK por el UA que originó el INVITE.

3.7.3. PROXY SIN MEMORIA DE ESTADOS

Estos servidores no almacenan ningún estado. Ellos reciben una petición, la reenvían al siguiente punto e inmediatamente borran todo lo relacionado con esa petición. Cuando este servidor recibe una respuesta el determina la ruta únicamente basado en el campo Via del encabezado y no mantiene ninguna información de esta transacción.

3.8. FORMATO DE LOS MENSAJES DE SIP

Una vez que se ha decidido que información deberá ser intercambiada entre sistemas, el siguiente paso es decidir como debe ser codificada esta información. Esta decisión tiene básicamente dos aproximaciones: binaria o basada en texto.

El protocolo SIP utiliza la codificación basada en texto, lo que ha creado una fuerte controversia, ya que los partidarios de esta idea proponen que su utilización en mas sencilla porque puede ser directamente leída por humanos, además de que los protocolos basados en texto son más flexibles y escalables; por otra parte, aquellos que defienden la codificación binaria, argumentan que este tipo de protocolos hacen un uso mas eficiente del ancho de banda y que también pueden ser escalables con las herramientas necesarias. Ambos tipo de codificación presentan ventajas y desventajas, sin embargo no hay que perder de vista que el protocolo SIP maneja codificación a base de mensajes de texto.

3.8.1. FORMATO DE LAS PETICIONES DE SIP

Una petición de SIP consiste en una línea de petición, varias líneas de encabezado, una línea vacía y un cuerpo de mensaje. El cuerpo del mensaje es opcional, algunas peticiones no lo traen.

Línea de petición: Esta línea contiene tres elementos: método, el Request-URI y la versión del protocolo. El método indica el tipo de petición. El Request-URI indica el siguiente salto, donde la petición tiene que ser enrutada. En la figura 3.14, el SIP proxy en softel.com recibe un INVITE con la Request-URI SIP:Lupita.Vazquez@softel.com. Este proxy sabe que Lupita puede ser localizada en dos direcciones por lo que genera dos INVITEs. Uno, contiene como Request-URI SIP:Lupita@unam.com y es enviado al servidor de unam.com. El segundo INVITE tendrá SIP:Lupita@131.160.1.112

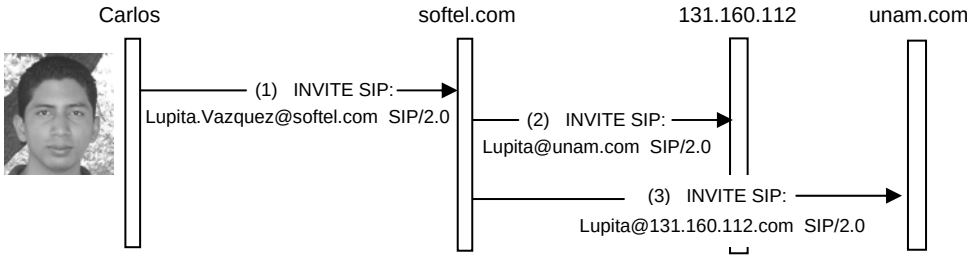


Figura 3.14 El Request-URI contiene el siguiente salto del camino

3.8.2. FORMATO DE LAS RESPUESTAS DE SIP

Una respuesta de SIP consiste en una línea de estado, varias líneas de encabezado, una línea vacía y un cuerpo de mensaje. El cuerpo del mensaje es opcional, algunas respuestas no lo traen.

Línea de estado: Esta línea contiene tres elementos: la versión del protocolo, el código de estado y la frase de estado. La versión actual del protocolo se escribe SIP/2.0. El código de estado reporta el estado de la transacción y la frase de estado que solo tiene sentido para ojos humanos, ya que las computadoras no procesan esta frase. A continuación se muestra un ejemplo de esta línea de estado.

SIP/2.0 180 Ringing

3.8.3. CABECERAS DE SIP

Las peticiones de SIP contienen los encabezados después de la línea de petición, mientras que las respuestas los ponen después de la línea de estado. Las cabeceras proveen información a cerca de los mensajes (ya sean peticiones o respuestas) y a cerca del cuerpo que contienen. Algunos encabezados pueden ser utilizados en ambos mensajes, pero otros son específicos ya sea para peticiones o respuestas únicamente. El encabezado consiste en el nombre del encabezado, seguido por dos puntos y el valor del encabezado.

Por ejemplo el encabezado llamado From, el cual identifica el origen de una petición particular se escribiría como sigue:

From: Lupita Vázquez <sip:Lupita.Vazquez@softel.com>

Como se puede ver en el ejemplo anterior, el valor del encabezado puede tener varios campos; en este ejemplo, el encabezado llamado From tiene dos campos: el nombre de la persona y su SIP URL.

La tabla 3.4 contiene los encabezados de SIP definidos por el protocolo.

Accept	Content-encoding	Max-forwards	Route
Accept-encoding	Content-language	MIME-version	Server
Accept-language	Content-length	Organization	Subject
Alert-info	Content-type	Priority	Supported
Allow	Cseq	Proxy-authenticate	Timestamp
Also	Date	Proxy-authorization	To
Authorization	Encryption	Proxy-require	Unsupported
Call-ID	Error-info	Record-route	User-agent
Call-info	Expires	Require	Via
Contact	From	Response-key	Warning
Content-disposition	In-reply-to	Retry-after	WWW-authenticate

Tabla 3.4 Encabezados de SIP

A continuación se explicará el propósito de los encabezados de SIP más importantes.

Call-ID. Este encabezado identifica una invitación en particular y todas las transacciones subsecuentes relacionadas a esta invitación. Este encabezado tiene el siguiente formato:

Call-ID: ges456fcdw211kfgte12ax@workstation1234.unam.com

Un servidor que maneja señalización de SIP para diferentes sesiones emplea el Call-ID para asociar los mensajes entrantes a la sesión adecuada.

Contact. El encabezado de Contact provee una URL donde el usuario puede ser contactado directamente. Esta característica es importante porque de esta manera los servidores SIP no necesitan estar en la trayectoria de los mensajes después de haber enrutado el primer INVITE.

Por ejemplo Lupita llama a Carlos a SIP:Carlos.Villanueva@softel.com, el servidor proxy de softel.com reenvía el INVITE a SIP:Carlos@131.160.1.112, donde se encuentra Carlos. Él acepta la llamada. El UA de Carlos responde un 200 OK con el siguiente encabezado de Contact:

Contact: Carlos Villanueva <sip:Carlos@131.160.1.112>

Cuando el UA de Lupita recibe este 200 OK, el manda un ACK directamente a SIP:Carlos@131.160.1.112, sin necesidad de atravesar el proxy de softel.com, ya que Carlos había sido previamente localizado. Las peticiones subsecuentes como el BYE son enviadas directamente entre los participantes de la sesión (Fig. 3.15).

Csec. El encabezado Csec (Command Secuence) tiene dos campos: un número entero y el nombre de un método. La parte numérica de este encabezado, es usada

para ordenar las diferentes peticiones durante la misma sesión (la cual esta definida por un mismo Call-ID). Es también utilizado para hacer corresponder peticiones con respuestas. Por ejemplo, Carlos manda un INVITE a Lupita con el siguiente Csec:

Csec: 1 INVITE

Lupita le responde un 200 OK con el mismo Csec que el del INVITE. Si Carlos quiere modificar la sesión que esta actualmente establecida, entonces puede mandar un segundo INVITE con el siguiente Csec:

Csec: 2 INVITE

Si una retransmisión del 200 OK es retrasada por la red y llega al UA de Carlos después de que él haya generado el segundo INVITE, sabrá que este OK es una respuesta al primer INVITE gracias al encabezado de Csec (Fig. 3.16).

Después de un INVITE las peticiones subsecuentes (excepto el ACK y el CANCEL) contienen un Csec que es el resultado de incrementar en uno al Csec de la petición original.

El Csec del mensaje de ACK es el mismo Csec que contiene el INVITE que el confirma. Para el CANCEL, el encabezado Csec es el mismo que tiene la petición que esta cancelando.

From. El encabezado From contiene el nombre del iniciador de la petición, así como su SIP URL:

From: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>

To El encabezado To contiene siempre la información del que recibe la petición. Usualmente contiene también la dirección pública del destino.

Via Este encabezado, almacena todos los proxies que manejan la petición, por lo tanto, contiene el camino tomado por el mensaje de petición (Fig. 3.17). Esta información es utilizada para enlutar las respuestas en dirección del cliente que generó la petición, además es útil para detectar lazos de ruteo, gracias a la información del encabezado de Via, cualquier proxy puede detectar si una petición es retransmitida en un lazo. Este encabezado se ve como se muestra a continuación:

Via: SIP / 2.0 / UDP workstation1234.softel.com

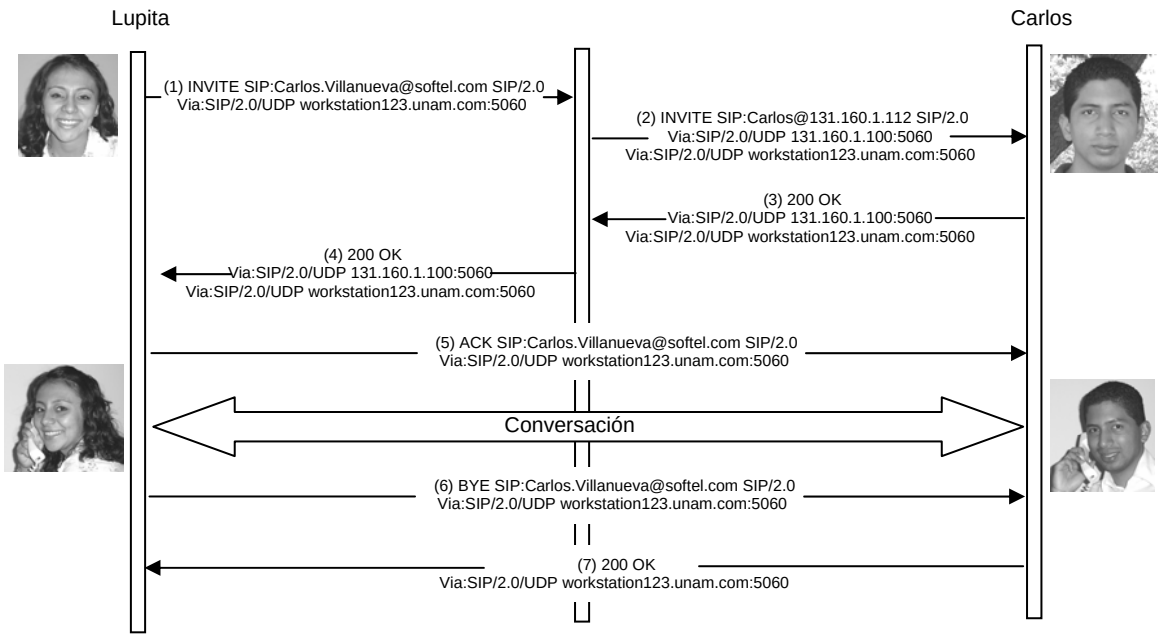


Figura 3.15 El encabezado de Vía almacena la ruta tomada por una petición

3.8.4. CUERPO DEL MENSAJE DE SIP

Tanto peticiones como respuestas pueden contener cuerpos de mensaje separados de los encabezados por una línea en blanco. El cuerpo de mensaje que llevan los paquetes de SIP es usualmente una descripción de sesión. Los SIP proxies no necesitan examinar el cuerpo del mensaje que reciben, esta información es transparente para ellos, por lo que las descripciones de sesión son transmitidas entre usuarios finales (UA's).

Así como un e-mail puede llevar varios archivos adjuntos, los mensajes de SIP pueden llevar varios cuerpos de mensaje. Por ejemplo, Lupita puede enviar un INVITE con dos cuerpos: una descripción de sesión y su foto, de esta forma el UA de Carlos puede mostrar la foto en la pantalla mientras que esta siendo alertado.

3.9. CAPA DE TRANSPORTE

SIP es un protocolo de la capa de aplicación, por lo que utiliza a los protocolos de la capa de transporte para transmitir tanto las peticiones como las respuestas. El comportamiento de cualquier protocolo da la capa de aplicación varía con el tipo de transporte utilizado. Si éste es confiable, el protocolo de la capa de aplicación arma un mensaje y lo manda a través de la capa de transporte con la intención de que éste llegue a su destino. La capa de aplicación no sabe como la capa de transporte logra el envío, solo sabe que la tarea es realizada.

¿Cómo se realiza? Típicamente la capa de transporte retransmite el mensaje hasta que el otro extremo lo recibe y manda un mensaje de confirmación. Estas retransmisiones son transparentes para la capa de aplicación.

Por otra parte, si un protocolo de la capa de aplicación corre sobre un protocolo de transporte poco confiable, como lo es UDP, este protocolo de aplicación no puede asumir que el

paquete se ha enviado. Por eso, se deben implementar retransmisiones en la capa de aplicación. Éstas se implementan comúnmente como sigue:

Los protocolos de la capa de aplicación forman mensajes y los pasan a la capa de transporte, si después de cierto tiempo no se ha recibido una confirmación de la recepción de este mensaje de parte del usuario destino, se tendrá que enviar nuevamente este mensaje.

3.10. EJEMPLO DETALLADO

A continuación se muestra un ejemplo detallado que nos permite esquematizar las transacciones del protocolo SIP.

LLAMADA A TRAVÉS DE UN SERVIDOR PROXY

En la figura 3.18, Lupita llama a Carlos a su dirección pública, pero el no se encuentra allí. Un servidor Proxy de softel.com enruta la llamada a su ubicación actual: SIP:Carlos@131.160.1.112. Este ejemplo consta de tres diferentes transacciones: INVITE, ACK y BYE.

INVITE DEL UA DE LUPITA AL SIP PROXY

```
INVITE sip:Carlos.Villanueva@softel.com SIP/2.0
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Lupita Vazquez <sip:Lupita@workstation123.unam.com>
Content-Type: aplication/sdp
Content-Length: 154

v=0
o=Lupita 2891234526 7864983648 IN IP4 workstation123.unam.com
s=Quiero saber de tí.
c=IN IP4 138.85.27.10
t=0 0
m=audio  20002  RTP/AVP  0
```

INVITE DEL SIP PROXY A CARLOS

```
INVITE sip:Carlos@131.160.1.112.com SIP/2.0
Via: SIP /2.0/ UDP 131.160.1.110
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Lupita Vazquez <sip:Lupita@workstation123.unam.com>
Content-Type: aplication/sdp
Content-Length: 154

v=0
o=Lupita 2891234526 7864983648 IN IP4 workstation123.unam.com
s=Quiero saber de tí.
c=IN IP4 138.85.27.10
t=0 0
m=audio  20002  RTP/AVP  0
```

RESPUESTA PROVISIONAL DE CARLOS AL PROXY

```
SIP/2.0  180  Ringing
Via: SIP /2.0/ UDP 131.160.1.110
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Carlos Villanueva < sip:Carlos@131.160.1.112 >
```

RESPUESTA PROVISIONAL DEL PROXY A LUPITA

```
SIP/2.0  180  Ringing
Via: SIP /2.0/ UDP 131.160.1.110
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Carlos Villanueva < sip:Carlos@131.160.1.112>
```

RESPUESTA FINAL DE CARLOS AL PROXY

```
SIP/2.0  200  OK
Via: SIP /2.0/ UDP 131.160.1.110
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Carlos Villanueva < sip:Carlos@131.160.1.112 >
Content-Type: aplication/sdp
Content-Length: 154

v=0
o=Carlos 2891234526 7864983648 IN IP4 131.160.1.112
s=Quiero saber de tí.
c=IN IP4 131.160.1.112
t=0 0
m=audio  41000  RTP/AVP  0
```

RESPUESTA FINAL DEL PROXY A LUPITA

```
SIP/2.0  200  OK
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Carlos Villanueva < sip:Carlos@131.160.1.112 >
Content-Type: aplication/sdp
Content-Length: 154

v=0
o=Carlos 2891234526 7864983648 IN IP4 131.160.1.112
s=Quiero saber de tí.
c=IN IP4 131.160.1.112
t=0 0
m=audio  41000  RTP/AVP  0
```

ACK DE LUPITA A CARLOS

```
ACK  sip:Carlos@131.160.1.112  SIP/2.0
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 1 INVITE
Contact: Lupita Vázquez < sip:Lupita @ workstation123.unam.com>
```

BYE DE LUPITA A CARLOS

```
BYE  sip:Carlos@131.160.1.112  SIP/2.0
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 2 BYE
Contact: Lupita Vázquez < sip:Lupita @ workstation123.unam.com>
```

RESPUESTA FINAL DE CARLOS A LUPITA

```
SIP/2.0  200  OK
Via: SIP /2.0/ UDP workstation123.unam.com:5060
From: Lupita Vázquez <sip:Lupita.Vazquez@unam.com>
To: Carlos Villanueva <sip:Carlos.Villanueva@softel.com>; tag=314159
Call-ID: 12345678@workstation123.unam.com
Cseq: 2 BYE
Contact: Carlos Villanueva < sip:Carlos@131.160.1.11 2 >
```

El ejemplo anterior muestra todos los encabezados que fueron explicados previamente trabajando de una manera conjunta para establecer una sesión de voz.

CAPÍTULO 4
IMPLEMENTACIÓN

4.1 ANTECEDENTES TEÓRICOS

4.1.1 PROTOCOLO FTP (*FILE TRANSFER PROTOCOL*)

FTP permite la transferencia de archivos entre dos *hosts* de la red. El protocolo diferencia entre una conexión de control y otra conexión de datos.

La conexión de control sigue la misma filosofía que las conexiones que se realizan para protocolos como SMTP, POP3 o NNTP, donde se envían comandos y se esperan respuestas que indiquen si la función solicitada se pudo llevar a cabo. La diferencia radica en que los datos no son enviados por esta conexión sino que se abre otra conexión de datos diferente para este fin.

La conexión de control, como las conexiones realizadas en otros protocolos, la realiza el cliente a la dirección IP del servidor y a un puerto conocido y estandarizado para este servicio, en este caso el protocolo FTP tiene asignado el puerto 21.

La forma de establecer la conexión de datos se negociará entre el cliente y el servidor, pudiendo comenzar dicha conexión cualquiera de los dos. En ambos casos se deberán indicar la dirección IP y el puerto al cual realizar dicha conexión.

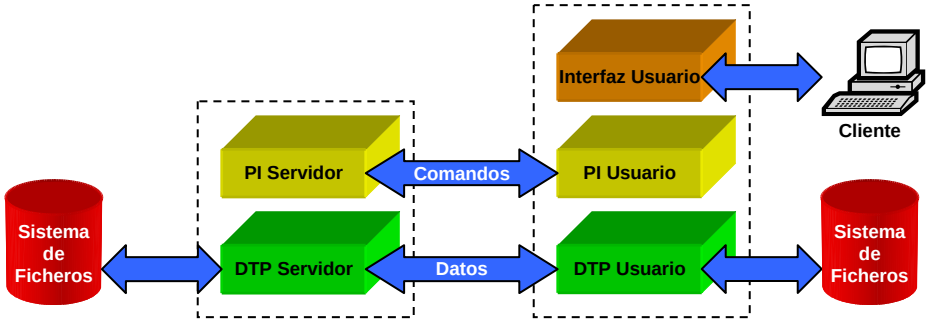


Figura 4.1. Conexión FTP

PI (*Protocol Interface*): Interfaz del programa que el usuario o el servidor maneja. Se encarga de manejar la conexión de control.

DTP (*Data Transfer Process*): Rutinas de transferencia propiamente dichas. Se encarga de establecer y manejar la conexión de datos.

El protocolo fue diseñado para ser independiente de las representaciones particulares que cada host de la red pudiera tener de los datos, entonces, es necesario que se indiquen al protocolo cuatro parámetros que establecerán la forma en que se van a representar los datos del archivo que se pretende transferir:

TIPO DEL ARCHIVO

El protocolo FTP distingue entre cuatro tipos diferentes de archivos:

- Local: permite que dos host que usan internamente tamaños de byte diferentes puedan intercambiarse archivos.
- Binario: indica que el archivo debe tratarse como un flujo de datos sin ningún formato interno específico.

- EBCDIC (*Extended Binary Coded Decimal Interchange Code*): permite la transferencia de archivos entre dos sistemas que usen internamente codificación EBCDIC. Este tipo se usará para la transferencia de archivos de texto.
- ASCII: está pensado para la transferencia de archivos de texto. El protocolo tomará ASCII como tipo de archivo por defecto.

FORMATO DEL ARCHIVO

FTP define tres formatos diferentes:

- Nonprint: formato por defecto para los archivos de texto.
- Telnet: Usa códigos para que las impresoras puedan formatear verticalmente el texto.
- Fortran: También hace uso de códigos especiales para formatear el texto.

ESTRUCTURA DEL ARCHIVO

Se definen las siguientes:

- Archivo: Se toma por defecto. No hay una estructura interna concreta definida, el archivo se considera un flujo de bytes continuo.
- Registro: el archivo se considera un conjunto de registros secuenciales.
- Página: permite el manejo de archivos que se estructuren en páginas independientes.

MODO DE TRANSMISIÓN

Se definen tres modos de transmisión en el protocolo FTP:

- *Block*: el archivo se divide en varios bloques, a cada uno de estos bloques se le añade una cabecera y cada bloque será enviado junto con su cabecera de forma ordenada a través de la conexión.
- *Compressed*: Usa un algoritmo RLE (*Run Length Encoding*) para comprimir los datos del archivo antes de ser enviados. Un algoritmo RLE es muy sencillo, consiste en sustituir una secuencia de caracteres iguales por el carácter y el número de veces que se repite.
- *Stream*: el archivo se interpreta como un flujo de bytes.

4.1.1.1 LA CONEXIÓN DE DATOS

La conexión de datos se establece para el envío de datos entre el cliente y el servidor.

Estos datos pueden ser:

- Listado de archivos o directorios que envía el servidor al cliente como respuesta a los comandos LIST o NLST.
- Archivo enviado desde el cliente al servidor.
- Archivo enviado desde el servidor al cliente.

Una vez establecida la conexión de control, como resultado de una conexión desde el cliente al puerto 21 del servidor, el cliente ya puede enviar a través de dicha conexión comandos al servidor. El servidor por su parte, responderá a estos

comandos con códigos de retorno, los cuales también son enviados a través de la conexión de control.

Llegará un momento en que el cliente necesitará obtener un listado de archivos, querrá enviar u obtener un archivo del servidor. Antes de que este intercambio de datos pueda comenzar, el cliente debe especificar al servidor con el comando PORT una dirección IP y un puerto a los cuales el servidor puede conectarse para efectuar la transferencia.

Una vez que el servidor conoce estos datos, el cliente crea un socket, le asocia la dirección IP y el puerto indicados, y se queda esperando a que el servidor se conecte a él, momento en el cual podrá comenzar el intercambio efectivo de datos.

Una vez completada la transferencia, el cliente cierra la conexión de datos. Hay que resaltar que la conexión de control sigue abierta mientras dure el diálogo entre cliente y servidor, en cambio, cada vez que se necesiten transferir datos, se creará una conexión de datos y una vez concluida la transferencia se cerrará.

El servicio FTP es uno de los más antiguos dentro de Internet. Es un servicio que los usuarios utilizan frecuentemente para compartir archivos.

El servicio FTP se puede realizar a través de los navegadores de Internet, así como a través de clientes FTP. En nuestro caso, lo que se va a explicar es la configuración de un Servidor FTP, que podrá ser accedido de cualquier forma.

Los servidores FTP controlan el acceso de los usuarios dentro de su sistema de carpetas. Esto quiere decir que, para conectarse a un servidor FTP, necesitamos un usuario (*login*) y una contraseña (*password*)**.** Se ofrece una alternativa de acceso no autentificado: el usuario anónimo (*anonymous*), que no lleva contraseña. Normalmente, este usuario tiene los permisos restringidos. No obstante, hay que destacar la responsabilidad del propietario del servidor FTP para crear y dar mantenimiento a la base de datos de usuarios y direcciones IP, se recomienda de forma importante, que se deshabilite o elimine al usuario *anonymous*, y que se creen los usuarios autenticados con *login* y *password*.

4.1.2 SOCKETS

Se denomina *socket* a un punto de comunicación a través del cual un proceso puede comunicarse con otro. Los procesos que intercambian información a través de *sockets* pueden pertenecer al mismo sistema o bien pueden encontrarse en sistemas diferentes.

Generalmente cuando se realiza una comunicación en una red TCP/IP participan dos procesos de sistemas diferentes que intercambian datos a través de la red. Cada uno de estos procesos (un proceso cliente y un proceso servidor) se encuentra en un extremo de la comunicación, considerándose un *socket* como la abstracción de este extremo.

Cuando se requiera un *socket* para obtener ese extremo de la comunicación a través del cual poder enviar datos, se recurrirá a las funciones facilitadas por el API (*Application Programming Interfase*) para tal fin, concretamente para crear un *socket* se hará uso de la función ***socket()***, obteniendo un descriptor que identificará nuestro *socket*, de igual forma que un descriptor de archivo devuelto por la función *open()* identifica un archivo que abramos desde nuestro programa.

4.1.2.1 DOMINIOS DE UN SOCKET

Los *sockets* nos van a permitir trabajar con varios dominios de comunicaciones. Un dominio define básicamente dos características:

- La familia de protocolos que estarán disponibles para arbitrar el intercambio de datos entre los dos *sockets*.
- El formato de las direcciones de red que se usarán para identificar ambos extremos de la comunicación.

Los dominios más comunes son los dos siguientes:

- Dominio UNIX: los *sockets* de este dominio son los locales al sistema donde son creados. Permiten la comunicaión interna entre dos procesos. Se identifican por las constantes PF_LOCAL, PF_UNIX (nombre BSD antiguo para PF_LOCAL) o PF_FILE (nombre POSIX para PF_LOCAL).
- Dominio Internet: los *sockets* de este dominio se identifican por la constante PF_INET posibilitando la comunicación de dos procesos a través de una red TCP/IP. Para ello las direcciones de los sockets de este dominio permiten especificar direcciones IP y puertos TCP/UDP.

Existen más dominios, restringidos a aplicaciones más específicas, algunos de estos dominios disponibles son los siguientes:

- Comunicaciones por radio AX.25(PF_AX.25)
- Protocolo IPX de Novell (PF_IPX)
- Protocolo AppleTalk (PF_APPLETALK)
- Comunicaciones X.25 (PF_X25)
- Protocolo IPv6 (PF_INET6)
- Otros: PF_DECNET, PF_BRIDGE, PF_NETROM, etc.

La definición de los dominios puede consultarse en <linux/socket.h>:

```
#define AF_UNSPEC       0
#define AF_UNIX         1       /*Unix domain sockets*/
#define AF_LOCAL        1       /*POSIX name for AF_UNIX*/
#define AF_INET         2       /*Internet IP Protocol*/
#define AF_AX25         3       /*Amateur Radio AX.25*/
#define AF_IPX          4       /*Novell IPX */
#define AF_APPLETALK    5       /*AppleTalk DDP*/
#define AF_NETROM       6       /*Amateur Radio NET/ROM*/
#define AF_BRIDGE       7       /*Multiprotocol bridge*/
#define AF_ATMPVC       8       /*ATM PVCs*/
#define AF_X25          9       /*Reserved for X.25 project*/
#define AF_INET6        10      /*IP version 6*/
#define AF_ROSE         11      /*Amateur Radio X.25 PLP*/
#define AF_DECnet       12      /*Reserved for DECnet project*/
#define AF_NETBEUI      13      /*Reserved for 802.2.LLC project*/
#define AF_SECURITY     14      /*Security callback pseudo AF*/
#define AF_pseudo_AF_KEY 15     /*PF_KEY key management API*/
#define AF_NETLINK      16
#define AF_ROUTE AF_NETLINK     /*Alias to emulate 4.4BSD*/
#define AF_PACKET       17      /*Packet family*/
#define AF_ASH          18      /*Ash*/
#define AF_ECONET       19      /*Acorn Econet*/
#define AF_ATMSVC       20      /*ATM SVCs*/
#define AF_SNA          22      /*Linux SNA Project*/
#define AF_IRDA         23      /*IRDA sockets*/
#define AF_MAX          32      /*For now*/
```

4.1.2.1.1 DOMINIO DE INTERNET

Muchas de las funciones que se usarán, como pueden ser ***connect()***, ***accept()*** o ***bind()*** toman como parámetro un puntero a una estructura genérica *sockaddr* que representa la dirección de un *socket*.

```
Struct sockaddr {
        unsigned short sa_family;    /*address family, AF_xxx */
        char sa_data[14];            /* 14 bytes of protocol address*/
};
```

Los 14 bytes del campo *sa_data* son interpretados de una manera u otra dependiendo del valor del campo sa_family, que identifica el dominio con el cual estamos trabajando. Como ya se dijo anteriormente, el dominio del *socket* definirá una forma concreta de representar las direcciones de red, es decir, definirá una familia de direcciones.

Para la familia de protocolos PF_INET se define la familia de direcciones AF_INET. Ambas constantes están definidas con el mismo valor (2) su semántica es diferente, ya que las constantes PF_XXXX (Protocol Family) representan una familia de protocolos (en nuestro caso de Internet serían los protocolos TCP Y UDP) y por otro lado las constantes AF_XXXX (Ardes Family) representan una familia de direcciones (en nuestro caso las direcciones IP).

La estructura *sockaddr_in* específica el formato de una dirección de *socket* para el dominio PF_INET.

```
#define _SOCK_SIZE_ 16    /*sizeof(struct sockaddr) */
struct in_addr{
        _u32 s_addr;
};

struct sockaddr_in {
        short int          sin_family;
        unsigned short int sin_port;
        struct in_addr     sin_addr;
        unsigned char      _pad[_SOCK_SIZE_ - sizeof(short int)-
sizeof(unsigned short int)-sizeof(struct in_addr)] ;
};

#define sin_zero   _pad
```

La dirección de un *socket* para Internet constará de:

- La definición, en el campo *sin_family*, de la familia de direcciones que usaremos. El valor que pondremos en este campo será AF_INET.
- La definición, en el campo *sin_port*, del puerto TCP/UDP al cual nos conectaremos o el que usaremos para recibir conexiones, dependiendo del caso.
- La definición, en el campo *sin_addr*, de la dirección de internet del host al cual nos conectaremos o del que esperamos recibir conexiones o datos.
- El campo *sin_zero*, es un campo de relleno, que servirá para adaptar la estructura *sockaddr_in* a la estructura genérica *sockaddr* que es la que se espera como parámetro en las funciones.

4.1.2.2 LOS TIPOS DE *SOCKET*

Características de la Comunicación

El tipo de un *socket* especificará ciertas características de la comunicación que se llevará a cabo a través del *socket.* Algunas características a tener en cuenta son las siguientes:

- **Comunicaciones orientadas a conexión:** Antes de comenzar el intercambio de datos se establece una conexión entre los dos extremos. Una vez realizada la conexión la información enviada por un extremo será recibida por el extremo contrario y viceversa. Esta asociación permanece hasta que la conexión se rompa.
- **Comunicación no orientada a conexión:** No se establece una conexión previa entre los dos extremos, sino que uno de ellos envía directamente un mensaje al otro extremo.
- **Fiabilidad de la transmisión:** Grado de confianza en que no se pierda información antes de llegar a su destinatario. Las comunicaciones orientadas a conexión tienen un grado de fiabilidad más alto que las orientadas a conexión.
- **Recepción ordenada de la información:** Especifica si los datos se recibirán en el mismo orden en el que fueron enviados. En comunicaciones orientadas a conexión los datos se recibirán en orden, en comunicaciones no orientadas a conexión no se puede garantizar la llegada ordenada de los datos.
- **Conocimiento en recepción del tamaño de los datos:** En comunicaciones orientadas a conexión se recibe un flujo continuo de datos, siendo responsabilidad de la aplicación el dividir este flujo de información en mensajes. En comunicaciones no orientadas a conexión se recibe un mensaje aislado, lo que permite el control del tamaño del mismo y su aislamiento respecto a otros mensajes que se puedan recibir.
- **Envío de datos urgentes:** Son los denominados datos fuera de flujo/banda. Esot s datos se pasarán directamente al nivel de aplicación y no están sujetos al control de flujo junto a los otros datos.

Las definiciones de los diferentes tipos de sockets disponibles son:

- **SOCK_STREAM:** Permite establecer una comunicación orientada a conexión, bidireccional, fiable y ordenada para el intercambio de un flujo de bytes sin limitación de tamaño. Admiten el uso de datos urgente (fuera de banda). El protocolo de transporte usado en el dominio de Internet para este tipo de *sockets* es TCP.
- **SOCK_DGRAM:** Permite establecer una comunicación no orientada a conexión, sin garantizar el orden de llegada de los datos ni la fiabilidad de la transmisión. Los datos se envían en datagramas con un tamaño máximo fijo. El protocolo de transporte usado es UDP.
- **SOCK_RAW:** Permite el acceso a los protocolos internos del sistema, saltándose la capa de transporte de la pila TCP/IP y accediendo directamente a los protocolos de más bajo nivel como puede ser ICMP. El proceso que intente abrir un *socket* de este tipo deberá tener permisos de superusuario para poder llevar a cabo esta operación con éxito. El uso de este tipo de *sockets* permite la implementación de nuevos protocolos.
- **SOCK_RDM:** Reliably Delivered Message Socket.
- **SOCK_SEQPACKET:** Permite crear una comunicación orientada a conexión bidireccional, fiable y ordenad, para el intercambio de datagramas de tamaño máximo fijo.

- **SOCK_PACKET:** Tipo de *socket* específico de Linux. Permite el manejo de paquetes en el nivel de usuario sin que las capas intermedias entre el nivel de aplicación y el físico del stack TCP/IP añada o elimine información de ellos.

4.1.2.3 MANEJO BÁSICO DE *SOCKETS*

4.1.2.3.1 FUNCIÓN *socket()*

```
#include <sys/types.h>
#include <sys/socket.>

int socket(int domain, int type, int protocol);
```

La función *socket()* nos permite crear un *socket*. La función devuelve el descriptor del *socket* o un valor de -1 en caso de error.

La familia de protocolos que se usará para la comunicación se especifíca en el parámetro *domain*. El tipo de *socket* se indica en el parámetro *type*. El último parámetro *protocol* indica el protocolo a usar dentro de la familia de protocolos indicada. Normalmente el valor de este parámetro será 0 para dejar que el sistema establezca el protocolo adecuado teniendo en cuenta la familia seleccionada y el tipo de socket indicado.

Las constantes que definen los protocolos pueden consultarse en el archivo de cabecera <netinet/in.h>:

```
enum{

        IPPROTO_IP          = 0,    /*Dummy protocol for TCP*/
        IPPROTO_ICMP        = 1,    /*Internet Control Message Protocol*/
        IPPROTO_IGMP        = 2,    /*Internet Group Management Protocol*/
        IPPROTO_IPIP        = 4,    /*IPIP tunnels*/
        IPPROTO_TCP         = 6,    /*Transmission Control Protocol */
        IPPROTO_EGP         = 8,    /*Exterior Gateway Protocol*/
        IPPROTO_PUP         = 12,   /*PUP protocol*/
        IPPROTO_UDP         = 17,   /*User Datagram Protocol*/
        IPPROTO_IDP         = 22,   /*XNS IDP Protocol*/
        IPPROTO_IPV6        = 41,   /*IPv6-in-IPv4 tunnelling  */
        IPPROTO_ICMPV6      = 58,   /*ICMPv6*/
        IPPROTO_IP          = 255,  /*Raw IP packets*/
        IPPROTO_MAX
};
```

Si la llamada a la función *socket()* devuelve -1, se puede obtener la causa exacta del problema consultado la variable *errno* que puede tomar los siguientes valores:

- **EPROTONOSUPPORT(93)[1]:** El tipo de protocolo indicado no está soportado dentro del dominio especificado.
- **EMFILE(24):** No hay espacio en la tabla de descriptores del proceso para crear una nueva entrada donde almacenar el descriptor del socket.
- **ENFILE(23):** La tabla de descriptores de archivos del sistema está llena.
- **EACCES(13):** No tenemos los permisos necesarios para crear un socket del tipo especificado.

[1] Definición de constantes en archivo de cabecera <asm/errno.h>

- **ENOBUFS(105):** No hay recursos suficientes para asignar un buffer al socket. Hasta que no se liberen recursos no podrá crearse el socket.

4.1.2.3.2 FUNCIÓN *bind()*

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

La función *bind()* asocia un "nombre" a un socket, es decir, asocia una dirección IP y un puerto al socket. Normalmente se usará para:

- Para asociar una dirección y un puerto a un servidor.
- Un cliente que establezca comunicaciones no orientadas a conexión asociará un nombre al socket para poder recibir las respuestas a los datagramas que envíe a un servidor.

El parámetro *sockfd* es el descriptor del socket, el parámetro *my_addr* es un puntero a la estructura genérica *sockaddr*. *Addrlen* es tamaño de la estructura *sockaddr*.
La función devuelve un valor de 0 si no hay error y un valor de -1 en caso contrario. Los posibles valores de *errno* en este caso son las siguientes:

- **EBADF(9)[2]:** El socket sockfd no es un descriptor válido.
- **EINVAL(22):** El socket sockfd ya tiene un nombre asociado.
- **EACCES(13):** No tenemos privilegios suficientes para asociar el nombre que hemos indicado al socket.
- **ENOTSOCK(88):** sockfd no es un descriptor de socket.
- **EADDRNOTAVAIL(99):** No se puede asignar el nombre especificado.
- **EADDRINUSE(98):** El nombre especificado ya está en uso.
- **ENOSR(63):** Falta de recursos.

4.1.2.3.3 FUNCIÓN *listen()*

```
#include <sys/socket.h>

int listen(int s, int backlog);
```

Esta función suele ser usada por los servidores orientados a conexión (servidores con sockets del tipo SOCK_STREAM o SOCK_SEQPACKET) para prepararse a recibir conexiones de clientes.

Esta función se debe llamar tras haber obtenido un descriptor de socket con la función *socket()* y tras haberle asignado un nombre con *bind()*. Tras llamar a *listen()* lo normal es llamar a la función *accept()*.

El primer parámetro es un descriptor de *socket* obtenido mediante la función *socket()*, el segundo parámetro es el número de solicitudes de conexión que se pueden retener en espera por el sistema hasta que el servidor hace una llamada a *accept()* para aceptar una de ellas. El número máximo de conexiones en espera es 128 y está representado por la constante SOMAXCONN.

[2] Definición de errores en archivo de cabecera <asm/errno.h>

```
#define SOMAXCONN   128
```

Si el número máximo de conexiones en espera ha sido alcanzado y un cliente intenta conectarse, no se dispondrá de sitio para almacenar su conexión y dejarle en espera, por lo que el cliente recibirá un error ECONNREFUSED.

Si la llamada se ejecuta con éxito obtendremos un valor 0. En caso de error obtendremos un valor de -1. Los valores que podremos obtener en *errno* serán los siguientes:

- **EBADF(9):** El argumento s no referencia a un descriptor válido.
- **ENOTSOCK(88):** El argumento s no es un socket.
- **EOPNOTSUPP(95):** El argumento s hace referencia a un socket que no soporta la operación *listen*.

4.1.2.3.4 FUNCIÓN *accept()*

```
#include <sys/types.h>
#include<sys/socket.h>

int accept(int s, struct sockaddr *addr, int *addrlen);
```

La función *accept()* nos permite aceptar una conexión realizada por un cliente a un servidor orientado a conexión. Esta función tomará la primera conexión que esté en la cola de espera backlog a la cual se hacía referencia en el segundo parámetro de la función *listen()* y nos devuelve un descriptor de socket a través del cual poder hacer uso de dicha conexión. Si no hubiese ninguna conexión en espera, la función *accept()* provoca que el proceso se quede bloqueado en espera de la llegada de alguna conexión.

El primer parámetro *s*, es el *socket* en el cual recibiremos las conexiones de los clientes.

El segundo parámetro, *addr*, es un puntero a una estructura *sockaddr_in* donde se recibirán los datos del cliente remoto que ha iniciado la conexión.

El tercer parámetro, addrlen, indicará el número de bytes que la función ha almacenado en el parámetro addr.

Si no hay error, el valor devuelto por la función será un nuevo descriptor de *socket* que hace referencia a la conexión iniciada por el cliente. En caso de haber algún error, obtendremos el valor de -1. Los posibles códigos de error de la variable *errno* son los siguientes:

- **EBADF(9):** El descriptor *s* no es válido.
- **ENOTSOCK(88):** El descriptor *s* no hace referencia a un *socket* sino a un archivo.
- **EOPNOTSUPP(95):** El *socket s* no es del tipo SOCK_STREAM.
- **EFAULT(14):** El argumento *addr* referencia a una zona de memoria donde no se puede escribir.
- **EWOULDBLOCK (11):** El *socket* ha sido marcado como no bloqueante y no hay ninguna petición de conexión en espera.

4.1.2.3.5 FUNCIÓN *connect()*

```
#include <sys/types.h>
#include<sys/socket.h>

int connect(int s, struct sockaddr *serv_addr, int addrlen);
```

La función *connect()* permite a un proceso cliente comenzar el proceso de conexión a un proceso servidor.

El primer parámetro *s* es un descriptor de socket obtenido tras llamr a la función *socket().*

El segundo parámetro *serv_addr* es un puntero a una estructura *sockaddr_in* donde se indica la dirección y el puerto al cual queremos conectarnos.

El tercer argumento *addrlen* es el tamaño de la estructura referenciada por *serv_addr.*

Normalmente la función *connect()* será invocada por un cliente que quiere conectarse a un servidor a través de un protocolo orientado a conexión. En este caso, *connect()* provocará un intercambio de mensajes entre el sistema local y remoto que culminará en el establecimiento de una conexión o con la devolución de un código de error. En caso de un protocolo no orientado a conexión no se intenta establecer conexión con un sistema remoto, sino que simplemente se almacenan los datos de dirección y puerto especificados en el argumento *serv_addr* para usarlos como referencia del socket identificado por el primer argumento *s*.

Si la conexión se establece con éxito, la función *connect()* devuelve un valor de 0, en caso contrario devuelve -1. Los posibles errores son:

- **EBADF(77):** El descriptor s no es correcto.
- **EFAULT(14):** La estructura *sockaddr* referenciada por el argumento *addr* está fuera del espacio de direcciones del proceso.
- **ENOTSOCK(88):** El descriptor s no está asociado con un *socket*.
- **EISCONN(106):** El *socket* ya está conectado.
- **ECONNREFUSED(111):** El servidor remoto rechazó la conexión.
- **ETIMEDOUT(110):** Se agotó el tiempo establecido para realizar la conexión.
- **ENETUNREACH(101):** No se puede alcanzar la red.
- **EADDRINUSE(98):** La dirección ya está en uso.
- **EINPROGRESS(115):** El *socket* está marcado como no bloqueante y la conexión no puede ser establecida inmediatamente.
- **EALREADY(114):** El socket está marcado como no bloqueante y un intento de conexión anterior aún no se ha completado.

4.1.2.4 ENVÍO Y RECEPCIÓN DE DATOS

4.1.2.4.1 FUNCIÓN *write()*

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

La función estándar *write()* permite enviar datos por un socket de tipo SOCK_STREAM.

El primer parámetro *fd* hace referencia al descriptor del socket. El segundo parámetro *buf* es un puntero al buffer donde se almacenan los datos a enviar. El tercer parámetro *count* especifica el número de bytes que se escribirán.

La función *write()* devolverá el número de bytes escritos, en caso de error se devolverá un -1, Los posibles valores de *errno* son los siguientes:

- **EBADDF(9):** El descriptor *fd* no es válido.
- **EINVAL(22):** El descriptor *fd* no permite la operación de escritura.
- **EFAULT(14):** El buffer *buf* no se encuentra dentro del espacio de direcciones del proceso.
- **EPIPE(32):** El otro extremo de la conexión está cerrado, por lo que los datos escritos no pueden ser leídos en el otro extremo.
- **EAGAIN(11):** Los datos no pueden ser escritos en este momento y el proceso ha elegido un modo de escritura no bloqueante.
- **EINTR(4):** Antes de que se pudiese realizar la escritura de los datos, la función fue interrumpida por una señal.
- **ENOSPC(28):** No se pueden almacenar los datos.
- **EIO(5):** Error de E/S a bajo nivel.

4.1.2.4.2 FUNCIÓN *read()*

```
#include <uniste.h>

ssize_t read(int fd, void *buf, size_t count);
```

La función estándar *read()* permite leer datos de un *socket* de tipo SOCK_STREAM.

El primer parámetro *fd* es el descriptor del socket, el segundo *buff* es un puntero al buffer donde se almacenarán los datos leídos. El número de bytes que se leerán se especifica en el tercer parámetro *count* .

La función devuelve el número de bytes leídos o un valor de -1 en caso de error. La variable *errno* puede tomar los siguientes valores:

- **EINTR(4):** La función se interrumpió por una señal antes de leer los datos.
- **EAGAIN(11):** No había datos disponibles y el socket se había configurado como no bloqueante por lo que devuelve este error en vez de quedarse el proceso bloqueado en espera de la llegada de datos al socket.
- **EIO(5):** Error de E/S de bajo nivel.
- **EISDIR(21):** El descriptor hace referencia a un directorio.
- **EBADF(9):** El descriptor no es válido o no acepta la función de lectura.

- **EINVAL(22):** El descriptor no es válido para realizar una función de lectura.

4.1.2.4.3 FUNCIÓN *sendto()*

```
#include <sys/types.h>
#include <sys/socket.h>

int sendto(int s, const void *msg, int len, unsigned int flags, const struct sockaddr *to, in tolen);
```

Permite el envío de datos a un socket de tipo SOCK_STREAM (orientado a conexión) o SOCK_DGRAM (no orientado a conexión).

Como primer parámetro se indica el descriptor del socket.

A continuación un puntero al buffer que contiene los datos a enviar, el número bytes a enviar se indica con el tercer parámetro *len.*

El parámetro *flags* permite especificar el modo de envío de los datos:

- Un valor de 0 hace que send() se comporte igual que write()
- Un valor de MSG_OOB indica que lo que se envía son datos urgentes.
- Un valor de MSG_DONTROUTE hace que se ignoren los mecanismos de encaminamiento que puedan estar establecidos en los protocolos de nivel inferior, enviando directamente el mensaje a la interfaz de red adecuada teniendo en cuenta la red a la que pertenezca la dirección IP de destino.

El parámetro *to* es una estructura de tipo *sockaddr* que deberá ser completada para informar de la dirección IP y puerto destinatarios del mensaje.

El tamaño de la estructura *sockaddr* se pasa como último parámetro en *tolen*.

La función devuelve el número de bytes enviados o un valor de -1 en caso de error. Los posibles valores de la variable *errno* son los siguientes:

- **EBADF(9):** El descriptor no es válido.
- **ENOTSOCK(88):** El descriptor no es un socket.
- **EFAULT(14):** El buffer de datos *buf* no está en el espacio de direccionamiento del proceso.
- **EMSGSIZE(90):** El tamaño del mensaje imposibilita que éste pueda ser enviado de una sola vez, y el socket ha sido configurado para realizar únicamente este tipo de envios.
- **EWOULDBLOCK:** El socket ha sido configurado como no bloqueante, pero la función debería bloquearse.
- **ENOBUFS(105):** No se ha podido reservar memoria para buffers internos necesarios para realizar la operación.

4.1.2.4.4 FUNCIÓN *recvfrom()*

```
#include <sys/types.h>
#include <sys/socket.h>

int recvfrom(int s, void *buf, int len, unsigned int flags, struct sockaddr *from, int fromlen);
```

Permite leer datos de un *socket*, ya sea de tipo SOCK_STREAM (orientado a conexión) o SOCK_DGRAM (no orientado a conexión).

El primer parámetro *s*, hace referencia al descriptor del socket.
El segundo parámetro *buf* es un puntero al buffer donde se almacenarán los datos leídos, el tamaño de este buffer se indica en el parámetro *len*.

El parámetro *flags* puede tomar los siguientes valores:

- **MSG_OOB:** Recepción da datos urgentes.
- **MSG_PEEK:** Permite acceder a los datos recibidos sin eliminarlos del buffer de entrada una vez que la lectura se ha realizado.
- **MSG_WAITALL:** Hace que la operación se bloquee hasta completar el buffer de lectura.

La estructura de tipo *sockaddr* referenciada por el puntero *from* es completada por la función con los datos del emisor del mensaje, en caso de estar leyendo información de un socket no orientado a conexión. Este parámetro puede ser NULL si el socket es orientado a conexión.

Y el campo *fromlen* almacenará el tamaño de la estructura *sockaddr*.

La función *recvfrom()* con un valor de NULL en el parámetro *from* se comportaría como una función *recv()*.

La función devuelve el número de bytes leídos, o un valor de -1 en caso de error. Los valores de la variable *errno* pueden ser los siguientes:

- **EBADF(9):** El descriptor de *socket* es inválido.
- **ENOTCONN(107):** El *socket* es del tipo SOCK_STREAM pero aún no se ha realizado la conexión.
- **ENOTSOCK(88):** El descriptor no corresponde a un *socket*.
- **EWOULDBLOCK:** El *socket* ha sido configurado como no bloqueante pero la operación de lectura debería bloquearse, ya que no hay datos disponibles. También se puede recibir este error si se ha establecido un *timeout* en recepción y éste ha expirado antes de recibir datos.
- **EINTR(4):** La operación de lectura fue interrumpida por una señal antes de poder realizar la lectura de los datos.
- **EFAULT(14):** El buffer *buf* está fuera del espacio de direccionamiento del proceso.

4.1.2.5 SECUENCIA TÍPICA DE LLAMADAS

TRANSFERENCIA ORIENTADA A CONEXIÓN

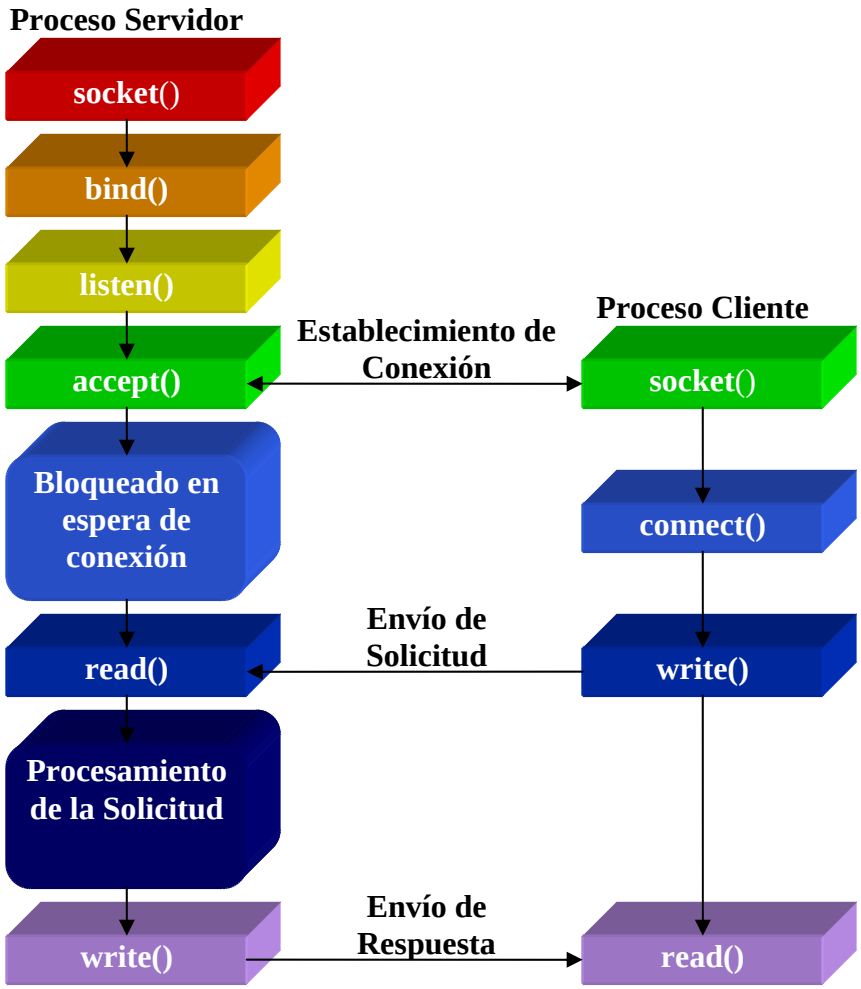


Figura 4.2. Transferencia Orientada a Conexión

TRANSFERENCIA NO ORIENTADA A CONEXIÓN

El proceso cliente no establece explícitamente una conexión con el servidor, sino que únicamente se limita a enviar a través de la red los datagramas que encapsulan los datos de la aplicación, indicando como destinatario de los mismos el proceso servidor. El servidor no necesita aceptar las peticiones de conexión del cliente, sino que se limita a recibir los datagramas que le vayan llegando.

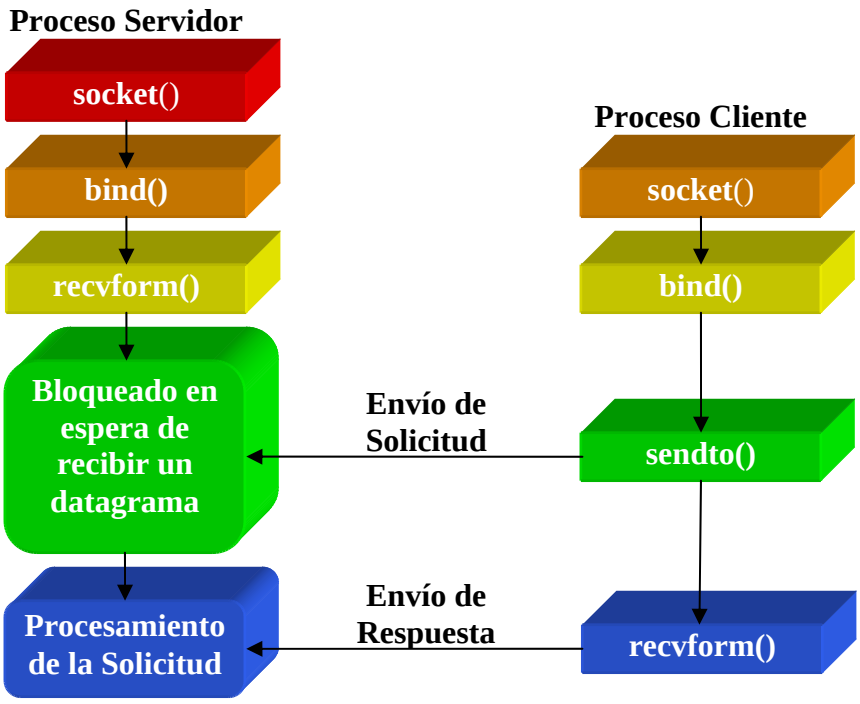


Figura 4.3. Transferencia No Orientada a Conexión

El tipo de socket que utiliza el protocolo SIP, como ya se mencionó es UDP (User Datagram Protocol).

4.2 ANÁLISIS DEL PROBLEMA

4.2.1 PLANTEAMIENTO DEL PROBLEMA

El proyecto se desarrolló en la empresa SOFTEL, S.A. de C.V., como parte de un programa de becarios.

La empresa Yaltel ubicada en la ciudad de Tijuana, Baja California requiere una aplicación que a través de Internet pueda establecer la comunicación tanto de voz como de datos entre un teléfono público y un servidor, mediante el cual la llamada pueda tener acceso a la PSTN (Public Switched Telephone Network), básicamente para establecer el servicio de llamadas a celulares y de larga distancia, ya que en llamadas locales esta compañía si tiene tarifas competitivas.

En esta aplicación, deben ser capturados cuatro dígitos de validación, los cuales, evitan el fraude a este tipo de teléfonos, ya que si alguien llega a cortar el cable e intenta utilizarlo como una línea telefónica normal, primero deberá enviar estos cuatro dígitos verificadores y con el tiempo determinado entre cada dígito.

Se tiene, además, que enviar una señalización al momento de establecer la llamada de tal manera que se pueda llevar a cabo una inversión de polaridad de –48 V a +48 V, lo cual permitirá al aparato telefónico realizar el cobro correspondiente a la llamada, esta

inversión, se realiza al momento que la persona a la que se llama contesta y se mantiene durante el transcurso de la llamada, pero es necesario mandar nuevamente la señalización al concluir la llamada para regresar la polaridad a su estado normal.

La aplicación se realizó utilizando el protocolo de comunicación SIP (Session Initiation Protocol), el cual se utiliza para establecer, mantener y terminar sesiones multimedia. Las sesiones multimedia incluyen telefonía en Internet, conferencias y otras aplicaciones similares que involucren sesiones media como audio, video y datos.

El protocolo SIP soporta sesiones unicast y multicast, así como llamadas punto a punto y multipunto.

La empresa GTEL en Tijuana, necesita un aplicación que permita que un teléfono publico se conecte a la caja donde se tiene la aplicación por un puerto FXS, y utilizando el protocolo SIP, se establezca la comunicación mediante Internet a un servidor ubicado en Tijuana, el cuál le dará la entrada a la PSTN (Public Switched Telephone Network),

4.2.2 REQUERIMIENTOS DE LA SOLUCIÓN

La empresa BBG nos envió el teléfono con el cual pretenden dar el servicio de telefonía pública y con el cual se debe enlazar a los usuarios a cualquier punto posible de la PSTN y la red celular.

La principal solicitud fue que se desarrollara un gateway con un número finito de puertos FXS a los cuales se les pudiera conectar teléfonos públicos de ciertas características y que se comunicaran con un servidor de SIP, que previamente la empresa había establecido, utilizando la red de datos IP y el protocolo SIP.

Las principales restricciones del uso del teléfono serían:

- Cuando se deposita dinero para marcar y se presiona el primer dígito, este mandará un código de protección *abcd* con un intervalo de tiempo específico entre cada letra para no ser imitado.
- Posteriormente a la recepción del código de protección, el gateway de Voz sobre IP debe ser capaz de validar o no el código y determinar si se brindará el servicio de telefonía o se trata de un intento de sabotaje para utilizar la conexión en forma ilegal.
- Al momento de establecer la llamada de voz, el teléfono depositará las monedas de cobro en su caja de seguridad e invertirá el voltaje de *-48 V* a *+48 V*, indicándole al gateway que la conexión tuvo éxito y se enlazará al usuario. El gateway deberá ser capaz de soportar ambas polaridades de voltaje e indicar debido a este cambio que la llamada fue exitosa.

Con la inversión de polaridad +48 V y -48 V y el código de protección abcd, se pretenden evitar los fraudes a la compañía.

Las principales características del gateway serán:

- Deberá enviar mensajes utilizando el protocolo SIP al servidor ubicado en Tijuana México para establecer una sesión, dicho servidor tendrá una cierta dirección IP pública asignada y permitirá o no a las peticiones de establecimiento de conexión acceder a la red telefónica pública.
- Enviará claves dentro de las peticiones de SIP que solo el servidor SIP entenderá.

- El formato del mensaje SIP deberá ser tal que el servidor lo reconozca como válido.
- Todas las peticiones de mensajes SIP enviados al servidor, al momento de intentar establecer una llamada de voz, deberán contener el campo Contact: ***Número_de_cuenta@dirección_ip_válida_de_salida***. El número de cuenta será proporcionado por la empresa.

Características del Servidor SIP:

- Se encargará de validar el formato del mensaje de petición de inicio de sesión y el número de cuenta contenido en el campo Contact: de la cabecera del mensaje.
- En caso satisfactorio, el servidor tomará el número telefónico enviado dentro del campo To: para establecer una llamada telefónica.
- Marcará el número nacional-internacional fijo o móvil desde un número telefónico de Tijuana válido para una central telefónica TELMEX, hacia cualquier punto de la red telefónica pública PSTN ya sea alámbrica o de la red celular.

4.2.3 CARACTERÍSTICAS IMPORTANTES DEL TELÉFONO PÚBLICO

Este teléfono es un equipo diseñado para ser instalado en recintos vigilados para uso público.

Presenta principalmente:

- Configuración de parámetros de acuerdo a las necesidades del operador.
- Facilidad de operación y de mantenimiento, tanto local como remoto.
- El teléfono TMI (Teléfono Modular de Interiores) forma parte del Sistema de Explotación de Teléfonos Modulares (SETM), que supervisa la operación, gestión y explotación de estos teléfonos. Este sistema realiza operaciones de identificación de los teléfonos, recepción y procesamiento de las alarmas y rutinas diarias enviados por el teléfono al SETM.
- El teléfono TMI tiene acceso total a la red, puede ser conectado a todos los tipos de centrales telefónicas de 48 V: Tipos rotary (excepto 7A1 y 7B1), barras cruzadas, semielectrónicas y electrónicas de conmutación espacial o temporal, cuyo sistema de marcación sea decádica o multifrecuencia.
- Se alimenta exclusivamente de la red no necesita ningún tipo de alimentación adicional.
- Los medios de pago permitidos en el teléfono son tarjetas prepago chip tipo EEPROM.

A través del display y del teclado el teléfono TMI proporciona diferentes funciones, tanto para el personal de mantenimiento como para el usuario.
Son las siguientes:

- Visualización y modificación de los parámetros locales.
- Visualización y borrado de las estadísticas.
- Pruebas funcionales de los principales elementos del TMI.
- Visualización de mensajes de aviso de averías y anomalías detectadas por las rutinas internas de supervisión.
- Llamadas y comunicaciones automáticas con el SETM.

IMPEDANCIA DE ENTRADA

El valor absoluto de la impedancia que presenta el teléfono con línea de 0 ohms y alimentación de 48 V.c.c y 2 x400 ohms, está comprendido en el rango de 600 a 900 ohms en la banda de 300 a 3 400 Hz.

NIVELES DE TRANSMISIÓN

El equipo no emite señales con valores de pico superiores a 1.5 V entre terminales de línea. Igualmente, en la condición de línea tomada, el valor eficaz de la potencia transmitida durante un periodo de 10 segundos no es superior a 100 µW (-10 dBm) cuando los terminales de línea se encuentren cargados con una impedancia de 600 ohms.

Finalmente, la emisión de señales codificadas (con excepción de los códigos de marcación multifrecuencia) presenta una potencia cuyo valor eficaz sobre cualquier periodo de duración 0.2 segundos es inferior a:

- 10 dBm para frecuencias de 300 a 3400 Hz.
- 33 dBm para frecuencias de 3.4 a 4.3 kHz.
- 37 dBm para f= 4.3 kHz, cayendo seguidamente 12 dB/octava hasta 16 kHz.
- 60 dBm para frecuencias superiores a 16 kHz.

DETECTOR DEL TONO DE INVITACIÓN A MARCAR

Margen de frecuencia: 425 Hz ± 75 Hz

Sensibilidad: 80 mVref mínimo

Duración: 300 ms. Mínimo

4.2.3.1 TELÉFONO PÚBLICO DE PRUEBAS

Para realizar pruebas con cualquier teléfono, es necesario conocer las siguientes características de transmisión particulares:

RESPUESTA EN FRECUENCIA

Representa la variación con la frecuencia de la amplitud de la señal en emisión, recepción o efecto local, para una excitación constante. Dicha respuesta condiciona las carácterísticas de transmisión, mejorando o empeorando la inteligibilidad.

DISTORSIÓN LINEAL

Los valores máximos de distorsión se indican a continuación:

- Distorsión armónica total (emisión y recepción): 10%
- Distorsión por armónicos impares: 4%
- Distorsión por intermodulación: 10%

EQUIVALENTES DE REFERENCIA

Los equivalentes de referencia se miden utilizando el procedimiento OREM-A de medidas objetivas con bucles de 0 a 1.200 Ohm. Para los equivalentes de emisión y recepción se fijan valores medios óptimos y sobre éstos unos límtes medios máximo y mínimo, que no deben ser sobrepasados.

Se admite, como máximo, un valor de desviación típica sobre la medida nominal cuyo valor es menor o igual, en emisión y recepción a 0.8 dB.

Se denomina SER al equivalente de referencia objetivo en emisión. Los valores límite máximo y mínimo de valores medios se fijan en:

SER(min)= -0.7 dB
SER(max)= +7.5 dB

Análogamente a lo expresado para el equivalente de referencia objetivo medio en emisión, se puede indicar para la recepción. Los límites se fijan en:

RRE(min)= 3 dB
RRE(max)= -1.7 dB

La ganancia de recepción adicional con el mando de regulación de volumen es de 6 a 10 dB.

Por último para el equivalente de referencia en efecto local, se dan valores superiores a:

STRE(0) > 5dB
STRE(1200) > 5 dB

CARACTERÍSTICAS DE LA MARCACIÓN

El aparato telefónico está equipado con los dispositivos de marcación necesarios para poder llevar a cabo tanto la marcación decádica, como la marcación por multifrecuencia.

Marcación Decádica

La marcación decádica se efectúa mediante una serie de aperturas de la línea, en la condición de descolgado (línea tomada).

Durante la marcación por impulsos, la condición de línea tomada debe satisfacer las características de c.c. establecidas anteriormente. El número de aperturas por dígito se corresponderá con el valor de dicho dígito. Sin embargo al dígito cero le corresponderán 10 aperturas. Para cada dígito, los impulsos de apertura se emiten con una frecuencia nominal de 10 ± 1 Hz.

El tiempo total de apertura será el 67% (± 3%) del tiempo total de un impulso.

- Tiempos Predígitos e Interdígitos

  El tiempo predigito es el tiempo que transcurre desde que se pasa de la condición del punto Resistencia en cc. a la del punto Resistencia durante los cierres de la marcación, hasta el comienzo del primer impulso de apertura del primer dígito de marcación, no es mayor que el tiempo interdígito que se define a continuación.

  El tiempo interdígito es el tiempo desde el final del último impulso de apertura hasta el comienzo del primer impulso de apertura del siguiente dígito. Mayor de 300 milisegundos.

- Resistencia Durante los Cierres de la Marcación por Impulsos

  La resistencia máxima es de 575, 300 y 76 ohms para corrientes de 10, 20 y 95 mA. En las condiciones límites nominales (puente 48V, 2 x 500 ohms y bucle 1200 ohms) es equivalente a decir que la corriente resultante no es inferior a 19 mA.

- Tiempo de Postdígito

  La condición de línea tomada por el sistema de marcación decimal se debe mantener más de 25 ms después de la última transición de un tren de impulsos. El límite superior es como máximo igual al tiempo interdígito.

- Corriente Durante Las Aperturas de la Marcación por Impulsos

  Durante los impulsos de apertura, cuando se aplique una tensión de 48 V.c.c. a los terminales de línea del equipo a través de 2 x 400 ohms, la corriente resultante no será superior a 0.25 mA.

Marcación Multifrecuencia

La marcación M.F. se realiza mediante la emisión de señales M.F. en un sistema multifrecuencia conforme a la Recomendación T/CS 45-02 de la CEPT.

- Codificación de los Dígitos

1209	1336	1477	Altas [Hz] / Bajas [Hz]
1	2	3	697
4	5	6	770
7	8	9	852
*	0	#	941

Las frecuencias de salida se mantienen dentro de ± 1.8% de su valor nominal para variaciones modulares de la impedancia de la línea desde 400 a 900 ohms.

- Niveles de Emisión de las Señales Multifrecuencia

Las señales de multifrecuencia se ajustan a los siguientes niveles de emisión medidos sobre una carga de 600 ohms:

Grupo de frecuencias altas: -6 dBm ± 2 dB
Grupo de frecuencias bajas: -8 dBm ± 2 dB
Preénfasis: 2 dB ± 1 dB

- Frecuencias Espúreas

Cuando una tecla esté actuada, el nivel de potencia total de todas las componentes de frecuencia espúreas, está al menos 20 dB por debajo del nivel de las componentes de frecuencia del grupo bajo de la señal. En cualquier caso, el nivel de cualquier componente individual de frecuencia espúrea no excede en la banda de frecuencias de 300 a 4300 Hz de -33 dBm.

- Tiempo de Subida

El nivel de cada una de las dos componentes, está por lo menos a 1 dB de su valor final en un tiempo inferior a 7 ms a partir del momento en que cualquier tecla haya sido actuada.

- Supresión de la Conversación.

Durante la emisión de secuencias de tonos de marcación se introduce una atenuación mínima de 60 dB en la vía de emisión del equipo telefónico asociado para evitar la transmisión de la palabra.

- Tono de Confianza

  A fin de que el abonado que llama tenga confirmación de la marcación multifrecuencia, percibirá los tonos multifrecuencia con un nivel agradable.

  El nivel del tono de confianza está comprendido entre 60 y 80 dB con desigualdad máxima de 10 dB entre los niveles de las dos frecuencias componentes.

ELEMENTOS EXTERNOS

El TMI es de construcción compacta, esto es, dispone de la seguridad necesaria de acuerdo al uso para el que ha sido diseñado. Es muy flexible de uso tanto para el usuario como para el personal de mantenimiento.

Vista Interior de Teléfono de Pruebas

Características en Corriente Contínua

- Polaridad

  Los parámetros de corriente contínua que van se especifican se satisfacen con independencia de la polaridad aplicada a las terminales de línea.

- Resistencia de Aislamiento

  En la condición de reposo, la resistencia de aislamiento entre las terminales de línea es mayor de 5 MΩ y entre las terminales de línea cortocircuitadas y tierra es mayor de 100 MΩ, medida con voltajes de hasta 100 V.c.c.

En la condición de línea tomada, la resistencia de aislamiento entre cualquiera de las terminales de línea y tierra es mayor de 100 MΩ, medida con voltajes de hasta 100 V.c.c

- Resistencia en c.c.

  En la condición de línea tomada, el equipo presenta entre las terminales de línea, una caída de voltaje máxima dada por la fórmula V= R*I + v, donde:

  R= valor resistivo equivalente presentado por el aparato telefónico.
  I= corriente de línea desde 18 mA hasta 100 mA.
  v= tensión de batería interna equivalente presentada por el aparato telefónico.

  Valores posibles para R y v son 150 Ωy 10 V respectivamente.
  En caso de marcación decádica, el valor anterior no se aplica durante el intervalo comprendido entre el primer impulso de apertura y 5 milisegundos después del último impulso de apertura para cada código.

  Dado que el equipo se alimenta exclusivamente a partir de la corriente suministrada por la línea telefónica a la cual está conectado, en la condición de colgado o reposo, se obtiene una corriente residual no superior a 1.5 mA, para mantener el contenido de memorias en el aparato telefónico.

- Límites de Corriente

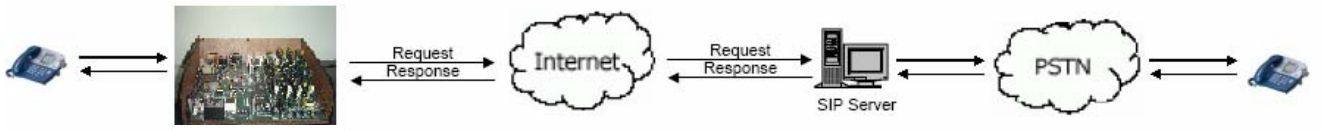
  Corriente Máxima

  El equipo en la condición de descolgado es capaz de soportar continuamente la alimentación correspondiente a 66V aplicados a través de una resistencia de 2 x 220 Ω.

  Corriente Mínima

  La corriente de línea mínima para que el aparato telefónico cumpla todo lo descrito será de 18 mA en la condición de línea tomada y dentro de la fase de conversación.

4.3 SOLUCIÓN PROPUESTA



Gateway SIP

Cada uno de los teléfonos públicos de la compañía será conectado a un gateway que contará con una serie de puertos FXS. El gateway controlará a cada uno de los teléfonos que sean conectados a él, proporcionándoles el servicio de voz y señalización de manera independiente.

El gateway se encargará de verificar cada una de las protecciones antifraude que use el teléfono, es decir, verificará la recepción del código *abcd* al iniciar la marcación en cualquier teléfono y se encargará de enviar la confirmación del cambio de polaridad en el teléfono *(+48 V -48 V)* para realizar el cobro por la llamada, cuando ésta se lleve a cabo con éxito.

La verificación del código antifraude *abcd,* se realizará revisando los datos que lleguen del teléfono al momento en que el usuario marque su primer dígito, es decir, se revisará cada una de las 4 letras y en caso de que el código no sea el correcto se enviará un mensaje de error de marcación y no se podrá establecer conexión alguna.

La inversión de polaridad de *-48 V* a *+48 V* se llevará a cabo con una adaptación hecha al gateway, de tal manera que ésta realice el cambio de voltaje al establecer la conexión de voz. Entonces, éste deberá enviar una señal, que puede ser un pulso, para indicar el cambio de polaridad.

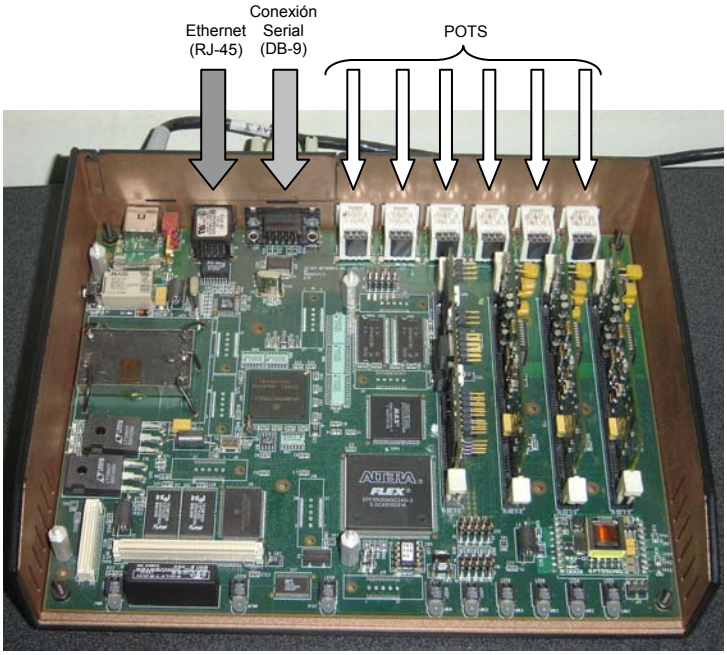
El gateway tendrá asignada una dirección IP pública o privada y se comunicará utilizando el protocolo SIP (Session Iniciation Protocol) en el puerto 5060 a un servidor SIP, ubicado en cualquier punto. Cada uno de los telefónos conectados a él, tendrá asignada la misma dirección IP de salida y lo que diferenciará a cada uno de los teléfonos será el puerto de voz utilizado.

Cada uno de los mensajes de SIP, tanto para las peticiones y respuestas, deberán seguir la norma RFC 2543 puesto que si los mensajes son diferentes entre el servidor y el gateway no podrá establecerse la sesión correctamente, pues la diferencia en los mensajes los hará no identificables por ambas partes.

Los mensajes enviados tendrán el formato necesario para ser identificados y se incluirán los números de cuenta para acceder al servidor SIP e ingresar a la red telefónica pública para establecer la conexión de voz. Como lo solicitó la empresa, los diversos números de cuenta de seguridad serán incluidos en el campo Contact: de los mensajes de petición para iniciar sesión SIP.

4.3.1 COMPONENTES DEL *GATEWAY*

HARDWARE



Vista Interior del Gateway SIP de Prueba

- 4 puertos FXS útiles.

- 1 puerto 10/100 Mbps ETHERNET
- 1 puerto serial con interfaz DB9
- 1 entrada para alimentación de 5 V DC

- 1 switch ON/OFF

SOFTWARE

- **VxWorks**

  Software con el cual se pueden llevar a cabo las aplicaciones que sean requeridas en el gateway. Las aplicaciones serán desarrolladas en lenguaje C y posteriormente se descargarán en la tarjeta madre del gateway. Solamente se podrá verificar la efectividad de las aplicaciones cuando el propio gateway esté funcionando.

  VxWorks es un sistema operativo multitarea, al igual que Linux, que trabaja en tiempo real pues el programador puede tener control sobre la manera en que son ejecutadas las tareas. Cada tarea tiene un rango de prioridad que va desde 0 (para la prioridad más alta) a 255 (para la prioridad más baja); supongamos que en un procesador se está ejecutando una tarea T1 con prioridad P1, si una tarea T2 con prioridad P2>P1 es llamada ya sea por el usuario o por una manejador de tareas, entonces T1 es suspendida y T2 es ejecutada. Si algún recurso no está disponible para ejecutar la tarea T2, ésta pasará a un estado de pendiente. Si la ejecución de T2 ha sido completada, T2 será descartada del itinerario y de cualquier manera el procesador regresará a T1 a menos de que T2 hubiera llamado a T3 con prioridad P3>P2>P1.

  Cuando dos tareas con la misma prioridad sean llamadas, VxWorks podrá usar la metodología “Round- Robin”, en donde el procesador dedicará alternativamente una cierta cantidad de tiempo a cada tarea hasta que éstas se completen.

  Entonces, el software VxWorks consiste de una o varias tareas que deben ser desarrolladas una computadora específica usando preferentemente un ambiente de desarrollo llamado Tornado. Una vez compilado y ligado, las tareas son cargadas en la memoria de la computadora VxWorks.

- **Tornado**

  Consiste de 3 componentes altamente integrados: el set de herramientas de Tornado, un grupo poderoso de herramientas de desarrollo y utilerías, el sistema VxWorks, un sistema operativo en tiempo real escalable de alto desarrollo que se ejecuta en el procesador objetivo y una amplia gama de opciones de comunicación (Ethernet, comunicación serial, emuladores, etc).

Originalmente Tornado fue desarrollado para resolver problemas inherentes a los ambientes de desarrollo cruzado, como por ejemplo la limitada comunicación entre el host y el objetivo, los limitados recursos del objetivo y las pobres herramientas integradas. El resultado fue:

- Una arquitectura que permite que todas las herramientas sean usadas en todos los objetivos, gracias a los recursos de los mecanismos de comunicación. Los componentes más importantes de esta arquitectura son: El servidor objetivo, que reside en la estación de trabajo del host, y el agente objetivo, que reside en la tarjeta del procesador objetivo.
- Un ambiente extenso y abierto que hace fácil integrar las 3 herramientas de desarrollo de hardware y software.
- Fácil desarrollo tanto para pequeños objetivos integrados o sistemas de multiprocesamiento de gran escala.

4.3.2 VENTAJAS Y DESVENTAJAS EN LA OPERACIÓN E IMPLEMENTACIÓN

La habilidad para dinámicamente leer y cargar los módulos objeto es importante para la arquitectura de Tornado: los desarrolladores no necesitan ligar la aplicación al kernel en el host ni bajar todo el ejecutable como un ambiente estático. Todos los módulos de aplicaciones pueden ser compartidos entre los miembros desarrolladores pero no necesitan ser re-ligados en el host. Esto hace posible para los desarrolladores agregar módulos objeto a un ambiente en vivo VxWorks cuando se hace un debugg o se reconfiguran las aplicaciones.

El gateway elegido generará los tonos: invitación a marcar, ocupado, ringback tone, de marcación de dígito. Además las funciones propias del gateway nos permitirán manipular los tiempos de marcación y de generación de los diferentes tonos.

La ventaja de utilizar este tipo de gateways, es la flexibilidad que puedan tener para el desarrollo de nuevas aplicaciones utilizando el software que tenga cargado. Estas nuevas aplicaciones podrán hacer la diferencia entre los diferentes sistemas telefónicos que estén utilizando la misma tarjeta madre y la propia tarjeta ofrecerá a los desarrolladores la posibilidad siempre latente de implementar estas nuevas aplicaciones si lo requirieran. Es importante señalar que todas las tareas podrán ejecutarse en el momento en que sean requeridas por los procesos generales del sistema o por otras tareas, es decir, se mantendrá un loop principal que será interrumpido cada vez que alguna tarea se inicialice y se regresará a él cada vez que las tareas concluyan.

Otra característica importante será que cada uno de los teléfonos conectados al gateway, mantendrá su funcionamiento independiente de los demás, en casi todos los aspectos: puerto de voz asignado, timers de marcación, generación de tonos, etc.

La desventaja es que al estar todos los teléfonos conectados a la misma tarjeta de voz, el procesamiento tendrá que ser el suficiente para soportar el funcionamiento del sistema y disminuir la probabilidad de que el sistema en general sufra algún incidente, ya que si en algún momento se presenta un error de ejecución, todos los teléfonos se verán afectados, en cuanto a que ya no podrán continuar con los ciclos y procesos en paralelo en un ciclo normal de intento de llamada.

Cuando alguno de las conexiones telefónicas sufre algún error de ejecución tanto en software o en hardware, no es posible reiniciar el funcionamiento de la conexión en

forma remota, es necesario reiniciar todo el gateway en un mismo instante perdiendo cualquier conexión establecida previamente.

De la misma manera, cuando sea necesario modificar el funcionamiento de las aplicaciones, será necesario utilizar la conexión serial y reiniciar el gateway en general.

VxWorks es un sistema operativo que no cuenta con un soporte de disponibilidad y eficiencia del mismo grado que las distribuciones de Linux, es decir es más probable que en algún momento el sistema sufra fallas generales o particulares en la ejecución de las funciones y que la puesta en marcha nuevamente requiera de mayor depuración de código.

Pero el princicpal problema con las aplicaciones del gateway, es que actualmente no se cuenta con una interfaz para que el usuario pueda acceder rápidamente a la configuración de cualquier característica del sistema, como: direcciones ip, puertos de voz, duración de timers para tonos utilizados, etc.

4.4 FUNCIONAMIENTO GENERAL

4.4.1 DESCRIPCIÓN DE LA CONFIGURACIÓN Y FUNCIONAMIENTO DEL *SOFTWARE*

El código fuente que se utilizo para la realización de la aplicación de Voz sobre IP, consta de varias procesos ejecutándose al mismo tiempo una vez que se inicializa el programa principal de la aplicación. Cada vez que el sistema se inicia, el programa principal revisa cada uno de los módulos principales en los que está subdividido según la siguiente figura:

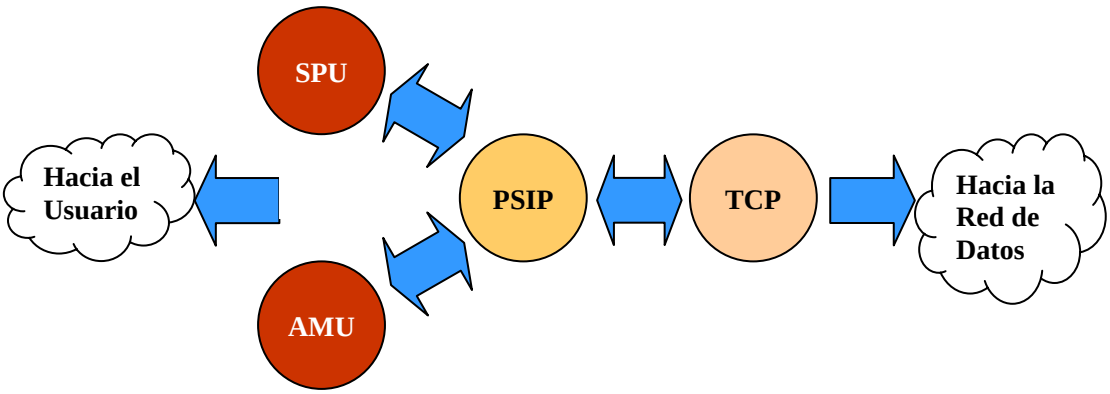


Figura 4.4. Funcionamiento del Software

Cada una de las aplicaciones desarrolladas en Visual C++ deben ser cargadas como imágenes del código fuente que se encuentra en carpetas internas del compilador, en este caso Gg6e. Gg6e consiste en una carpeta ubicada en el disco duro de la máquina que servirá como host y que contiene subdirectorios propios de la tarjeta, cuando se accede a la carpeta Gg6e\micro\basica\pform\sample se encuentran los archivos que forman una imagen que se cargará en la memoria de la tarjeta y que se ejecuta cuando ésta se enciende, por lo tanto será necesario modificarlos para conseguir que la tarjeta trabaje de una manera específica.

Los archivos de Gg6e\micro\basica\pform\sample\root son:

- ggroot.c
- kbtelamu.c
- kbtelapi.c
- kbtelcr2.c
- kbtelnet.c
- kbtelpp.c
- kbtelserv.c
- kbtelsh.c
- kbtelsocket.c
- kbtelspu.c
- kbteltcp.c

Los cuales se colocaron en un proyecto en común para hacerlos manejables. Estos archivos además de conformar la imagen que será descargada, corresponden a las etapas en las que está subdividido el sistema, según la figura anterior.

Cada uno de los archivos anteriores contiene funciones que corresponden a las etapas AMU, SPU, PSIP, TCP que a continuación se explican:

AMU y SPU: Aquí se encuentra la interfaz con el teléfono o con la tarjeta de Voz sobre IP, pues se controla para cada uno de los 4 teléfonos:
- La generación del tono ocupado (BUSY), en espera (DIAL) y en llamada.
- El colgado ONHOOK y descolgado OFFHOOK del teléfono.
- La decodificación de los dígitos marcados en el teclado del teléfono.
- El almacenamiento de los dígitos marcados en el teléfono.
- La verificación del código antifraude a-b-c-d.
- El envío de BYE cuando se cuelga el teléfono para terminar la sesión SIP.
- Se inician contadores de tiempo para indicar la duración de cada uno de los tonos.

PSIP: Aquí se encuentran las funciones que forman los mensajes de SIP con los que se establecen las sesiones, tales mensajes son tanto para las peticiones y las respuestas. Las peticiones corresponden a los mensajes INVITE (incluyendo la información que indica el tipo de sesión a iniciar SDP), BYE, CANCEL, ACK y REGISTER.
Las respuestas corresponden a distintos códigos entre ellos RINGING, TRYING, OK.

TCP: Aquí se encuentra el loop principal del sistema, pues en este lazo se encuentra siempre el sistema hasta que es interrumpido por algún otro proceso que se activa en paralelo pero, nunca deja totalmente este lazo principal. La configuración que se realiza en los teléfonos cada vez que inicia el sistema es la siguiente:
- Se inicializan los servicios de red y las variables en cada una de las estructuras.
- Se abre el socket UDP en el puerto 5060, por donde se enviarán y recibirán los mensajes SIP.
- Se asigna un número de identificación (TCID) y un puerto de voz por el que se enviará la voz a cada teléfono.

Loop Principal:
- Se hace una verificación de cada uno de los contadores de tiempo, si es que fueron asignados, por cada uno de los teléfonos.
- Se especifican acciones a tomar en los casos donde los contadores de tiempo se venzan.

- Se detecta la llegada de cadenas de datos al socket UDP y se identifica a qué tipo de mensajes corresponden.

Entre cada una de las etapas del sistema existen funciones de conexión, que les permiten interactuar entre ellas, ya sea en dirección de la red de datos desde la tarjeta o viceversa.

DESCARGA DE IMÁGENES

Cada vez que se requiera descargar una imagen debido a la modificación de cualquiera de los archivos arriba mencionados, se requiere:

- Guardar los cambios de los archivos en *Gg6e\micro\basica\pform\sample*, puesto que desde aquí se formará la nueva imagen.
- Abrir una terminal en MSDOS y cambiar el directorio actual a:
  C: Gg6e\micro\basica\pform\gg6e\micro\vxworks\gnu>
- Se establece el ambiente para generar la imagen con:
  C: Gg6e\micro\basica\pform\gg6e\micro\vxworks\gnu> setenv
- El archivo MAKEFILE compilará los archivos de sample, mostrando los posibles errores de código en alguno de dichos archivos si es que los hubiera. Para seguir adelante, será necesario revisar la línea de código y el archivo donde se indica el error. Este archivo generará la nueva imagen que se descargará utilizando el protocolo FTP.
  C: Gg6e\micro\basica\pform\gg6e\micro\vxworks\gnu> make ggbasica_all GG_LINK_DSP_IMAGE=1
- Es necesario levantar un servidor FTP en la computadora origen para que por esta vía se descargue la imagen.
- Al encender la tarjeta la imagen se descargará vía FTP y se comunicará vía ETHERNET con el resto de la red de información para el envío de los paquetes TCP y UDP.
- Al terminar de descargar la imagen, la tarjeta se mantendrá en los estados indicados por el código que fue descargado.

CONFIGURACIÓN GATEWAY- TARJETA

La tarjeta de Voz sobre IP necesita ser configurada para que puedan ser descargadas correctamente las imágenes utilizando el servidor FTP y pueda recibir paquetes UDP y TCP.

La tarjeta debe ser configurada por lo menos una vez, ya que los cambios se guardan aún apagándola.

La configuración se realiza como sigue:

- Es necesario tener disponible un puerto serial de la computadora desde donde se descargarán las imágenes vía FTP y en la cual residen los archivos de C: Gg6e\micro\basica\pform\sample. El otro extremo de la conexión llegará al puerto serial de la tarjeta de Voz sobre IP.
- Al llegar a este punto será necesario iniciar un servidor FTP en la máquina origen para que se lleve a cabo la descarga, configurando una cuenta de usuario y una contraseña, mismas que deberán ser iguales que las de la configuración de la tarjeta en el gateway.
- La interfaz entre la tarjeta y el usuario será por medio de la hyperterminal de Windows, desde donde se podrán observar los datos que sean recibidos en el puerto serial y que hayan sido enviados a imprimir en la imagen del código fuente descargado. La hyperterminal debe configurarse para que la velocidad de

transmisión sea la adecuada (9600 kbps) y los datos no se pierdan, además se especifica el puerto serial en donde recibirá los datos, generalmente COM1.

- Al encender la tarjeta se empieza a cargar VxWorks como arranque de la misma, dándonos la opción de configuración en el boot mediante un conteo regresivo.
- Para iniciar la configuración presionamos una tecla y poder iniciar utilizando el boot, de esta manera escribimos la letra c (minúscula ya que C mayúscula significa clear) que significa configuration.
- En la pantalla aparecerá:

  Host name: Nombre de la máquina en server.
  File name:
  Opciones
  - Flash (cargados por default).
  - C: Gg6e\micro\basica\pform\sample\root (archivos modificables arriba mencionados)

  Inet en Ethernet: IP de la Tarjeta : máscara de subred (opcional)
  Host Inet: IP del FTP server (máquina a la que se conecta)
  Gateway: IP Gateway por default.
  User: Usuario de cuenta FTP (Mismo que en el servidor FTP)
  Password:

- Para salir de boot: @

Inmediatamente después de la configuración se inicia el sistema operativo VxWorks con el cual se implementaran las funciones de la imagen descargada y que están compuestas por el código fuente de los archivos presentes en C: Gg6e\micro\basica\pform\sample.

4.4.2 DIAGRAMAS DE ESTADO Y DIAGRAMAS DE TIEMPO

VER ANEXO

4.4.3 EXPLICACIÓN DEL CÓDIGO FUENTE

A continuación iniciaremos la descripción del código fuente en C++ que se desarrolló para hacer las aplicaciones y características que el proyecto necesitaba. Nos referiremos a los archivos ubicados en en C: Gg6e\micro\basica\pform\sample\root y el archivo kbtel.h

Cada uno de los prototipos de las funciones, las estructuras y las constantes utilizadas en todo el proyecto de Visual C++ se encuentran declaradas en el archivo kbtel.h, el cual es requerido al iniciar cada uno de los archivos de los que constará el proyecto en general. Este tipo de declaración facilita el uso de todos los archivos, pues ya no será necesario hacer la declaración explícita del contenido de kbtel.h, por el contrario bastará con indicar **#include <kbtel.h>**.

4.4.3.1 kbtel.h

```
*****************************************************************************************
FUNCIONES ASOCIADAS PARA LA APLICACION
*****************************************************************************************
#ifndef __KBTEL_H__
#define __KBTEL_H__

#define KBTEL_MSG_FLASH_DIGIT      32

#include "vxWorks.h"
#include "stdioLib.h"
#include "strLib.h"
#include "sockLib.h"
#include "inetLib.h"
#include "hostLib.h"
#include "ioLib.h"
#include "selectLib.h"
#include "stdlib.h"

#include <Sockets.h>   /*Libreria con los prototipos  para manejar sockets */
#include <Drv_def.h>
#include <Rdrv.h>
#include <misc.h>

#include "c:/gg6e/micro/basica/common/micro/iface/amuapi.h"
#include "c:/gg6e/micro/basica/common/micro/npm/qsmu/qsmuinc.h"
#include "c:/gg6e/micro/basica/common/micro/npm/qsmu/qsmuloc.h"
#include "c:/gg6e/micro/basica/common/micro/tsg/spu/spuinc.h"
#include "c:/gg6e/micro/basica/pform/sample/nmm/nmminc.h"
SOCKET SipSocket;     /*Declaración del nombre y tipo del socket a utilizar*/
SINT32 kbtel_create(void);
```

En esta primera parte, se incluyen las librerías más importantes que se usarán a lo largo de todo el código, pues contendrán los prototipos de aquellas funciones utilizadas, declaraciones de estructuras, variables o constantes globales. Algunas de las tareas que se podrán realizar con la inclusión de estas librerías serán:

- Implementar el sistema operativo en tiempo real VxWorks y todas sus funciones.
- Realizar impresiones de variables o constantes y realizar operaciones aritméticas con las mismas.
- Abrir o cerrar los sockets de comunicación para el envío o recepción de datos. Este socket de nombre SipSocket podrá ser de tipo TCP o UDP.

```
*****************************************************************************************
DATA DEFINITION: Estructura teléfono
*****************************************************************************************
DESCRIPTION: Captura los dígitos marcados
*****************************************************************************************
int i;
typedef struct
{
        char digitos[30];
        int indice;
        int ind;
        char string2[30];
}numero;

numero telefono[5];
```

Esta estructura es una de las más importantes para el funcionamiento en general del sistema, pues aquí se organizarán las características de los 6 teléfonos disponibles en la tarjeta, las cuales son:

- Los dígitos marcados por cada uno de los 6 teléfonos *digitos[30]*, teniendo un máximo de 31 dígitos posibles.
- Los contadores de dígitos marcados *indice* e *ind* por cada uno de los 6 teléfonos.
- La cadena *string2[30]* que es donde se almacenará todo el número marcado para ser enviado a quien lo necesite, por cada uno de los 6 teléfonos.

Para llamar a cualquier característica de la estructura se hace referencia de la siguiente manera: *telefono[posición].característica*

```
*************************************************************************************************
DATA DEFINITION: Estructura header
*************************************************************************************************
DESCRIPTION: Guarda los elementos de la cabecera del mensaje SIP de cada teléfono
*************************************************************************************************
#define MAX_POTS            5

typedef struct
{
        unsigned char *callid;          /*Identificador de la transacción sip*/
        unsigned char *para;            /*Contenido del header To del mensaje*/
        unsigned char *desde;           /*Contenido del header From del mensaje*/
        unsigned char *via;             /*Contenido del header via*/
        unsigned char *tagdesde;        /*Identifica el tag que viene en el from del request*/
        unsigned char *tagpara;         /*Identifica el tag que contendrán los responses a ese
request*/
        unsigned char *method;          /*Método sip que inició la transacción*/
        unsigned char *last_message;    /*Ultimo mensaje recibido o generado de dicha transacción*/
        unsigned char port[5];          /*Puerto RTP que se obtiene del OK recibido*/
}HEADER;

HEADER header[MAX_POTS];
```

En esta estructura se definen los apuntadores (direcciones de memoria) en donde se guardará el contenido de cada uno de los mensajes SIP específicamente, para cada uno de los 6 teléfonos disponibles en la tarjeta. En cada declaración se describe qué se guardará en cada una de las direcciones de memoria.

En la estructura se declara el *port[5]*, en donde se guardará la cadena de números capturados del mensaje SIP OK recibido como respuesta a las peticiones SIP iniciales. Este puerto es de vital importancia pues si no se conoce su valor, no se podrá establecer la sesión de voz entre los usuarios, así los paquetes de voz no le serán enviados al puerto correcto a la máquina destino.

MAX_POTS será una constante global y será referida al número máximo de teléfonos disponibles.

Para llamar a cualquier característica de la estructura: *header[posición].característica*

```
*************************************************************************************************
DATA DEFINITION: Configuraciones fijas
*************************************************************************************************
DESCRIPTION: Guarda los datos que se requieren configurar de una manera fija
*************************************************************************************************
unsigned short FLAG;
```

```
typedef struct
{
        unsigned char ip_local[20];          /*IP que tiene el equipo local*/
        unsigned char ip_server[20];         /*IP servidor FTP */
        unsigned char cuenta[11];            /* Cuenta que permite conexión con PSTN*/
        unsigned char proxy[20];             /* Servidor Proxy de la empresa YALTEL*/
}CONFIGURACION;

CONFIGURACION config;
```

En esta estructura se declaran a las variables que contendrán las cadenas de caracteres que conformarán las direcciones IP tanto de la tarjeta del gateway *ip_local[20]* y del servidor proxy por medio del cual se accederá al servidor SIP de la compañía Yaltel *proxy[20]* o de algún destino específico. Se podrá configurar el contenido de ambas variables con las direcciones IP correspondientes y como son globales, serán válidas en todo el código.

La *cuenta[20]* contendrá el número de cuenta proporcionado por la compañía Yaltel, que se enviará en los mensajes SIP y se validará por el servidor SIP para brindar el servicio de voz. Si el número no se envía o es erróneo, no se completará la conexión entre los usuarios satisfactoriamente.

```
*****************************************************************************************
DATA DEFINITION: Estructura General del Mensaje
*****************************************************************************************
DESCRIPTION: Define la estructura interna general del mensaje
*****************************************************************************************

typedef struct
{
   /* parameters */
   DATA32 param1;
   DATA32 param2;
   DATA32 param3;
   DATA32 param4;
} KBTEL_MSG_GEN_BODY_T;
```

Cada uno de los parámetros de esta estructura manejan datos que son capturados por el sistema de la tarjeta cuando recibe algún tipo de mensaje en las funciones de interfase con los diferentes módulos internos, este mensaje recibido estará conformado por los 4 parámetros DATA32, que corresponden a un tipo especial de formato de dato que la tarjeta maneja. Cada uno de los 4 datos contendrá información útil como por ejemplo: *Param2* indicará si se ha recibido algún dígito marcado o en general si la tarjeta ha capturado datos en el momento en que *param2* sea revisado. *Param1* contendrá el dígito marcado o en general el dato capturado al momento de llamar a *Param1*, este parámetro será muy importante ya que de aquí se tomarán los dígitos que el usuario haya presionado para, posteriormente, formar el número telefónico con el cual se establecerá la llamada de voz. Este parámetro también nos indicará características importantes como si el teléfono se encuentra colgado o no y el tipo de tono a generar.

El resto de los parámetros *Param3* y *Param4* no se utilizarán.

```
*****************************************************************************************
DATA DEFINITION: Estructura Total del Mensaje
*****************************************************************************************
DESCRIPTION: Define la estructura interna total del mensaje
*****************************************************************************************
typedef struct
{
        UINT16 if_id;                        /* Interface ID */
```

```
        UINT16 call_id;                 /* "call" within interface */
        TCID tcid;
} KBTEL_MSG_HEADER_T;
```

En esta estructura, la variable *tcid* es la más importante, debido a que ésta nos indicará cuál de los 6 teléfonos activos, cada uno en su puerto, está siendo utilizado en cualquiera de las tareas que el software de la tarjeta es capaz de realizar. Esta variable *tcid*, controlará a la mayor parte de las estructuras que estén relacionadas con la generación de tonos, captura de dígitos marcados, direccionamiento de señalización o voz para cada uno de los teléfonos. Gracias a esta variable, se evitará el grave problema de direccionar mal las tareas y la señalización, pues así cada puerto telefónico recibirá lo que está esperando y enviará lo que debe enviar.

La variable *call_id* representará a los estados en que se encuentre cualquiera de los teléfonos, de estos estados el más importante será LOCAL_HOOK pues aquí se encuentran las tareas de llamada en espera, llamada conectada, teléfono colgado, teléfono descolgado y la generación de tonos como el de llamada en espera; así como también la tarea de capturar los dígitos marcados y formar el número telefónico.

```
*****************************************************************************************
DATA DEFINITION: Estructura Global del Mensaje
*****************************************************************************************
DESCRIPTION: Agrupa la estructura interna general y total
*****************************************************************************************

typedef struct
{
   KBTEL_MSG_HEADER_T header;
   KBTEL_MSG_BODY_T body;
} KBTEL_MSG_T;
```

Header y *body* contendrán a las variables del mensaje definidas anteriormente en la estructura interna general y total del mensaje. Es importante señalar que el mensaje en general está definido para el control interno y la comunicación entre cada una de las funciones propias del sistema y así poder llevar a cabo cada una de las tareas.

```
*****************************************************************************************
DATA DEFINITION: Tipos de Timer
*****************************************************************************************
DESCRIPTION: Define los tipos de timer por teléfono (POT)
*****************************************************************************************

MX_TimerId TimerPots[MAX_POTS];              /*Timer pot marcación de dígitos*/
MX_TimerId TimerDial[MAX_POTS];              /*Timer pot llamada en espera*/
MX_TimerId TimerBussy[MAX_POTS];             /*Timer pot  tono ocupado*/
MX_TimerState TimerPotst[MAX_POTS];          /*Estado del timer marcación de dígitos*/
MX_TimerState TimerDialst[MAX_POTS];         /*Estado del timer llamada en espera*/
MX_TimerState TimerBussyst[MAX_POTS];           /*Estado del timer tono ocupado*/

#define IDLE 12     /*12 seg llamada en espera*/
#define BUSY 10    /* 10 seg tono ocupado*/
```

```
*****************************************************************************************
PROTOTIPOS DE FUNCIONES KBTEL
*****************************************************************************************

extern unsigned short create_request(TCID tcid,char *method,char *sdp,char *id);
```

USO: FUNCIÓN DONDE SE CREAN LAS PETICIONES DE MENSAJES SIP

PARÁMETROS DE ENTRADA:

tcid	Identificador del teléfono (Pot) 0…4
method	Indica la petición SIP
sdp	Contenido de la petición SIP

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:
Ninguno.

EVENTOS:
Envía datos por el socket SOCK_SendTo

```
extern unsigned long ConvStrToLong(char *StrIP);
```

USO: FUNCIÓN QUE CONVIERTE CADENAS DE CARACTERES A ENTEROS

PARÁMETROS DE ENTRADA:

StrIP	Cadena de caracteres

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:

LongIP	Cadena de enteros

EVENTOS:
Ninguno

```
extern char *create_sdp(unsigned char *ip, unsigned short port,unsigned short codec,unsigned short tcid);
```

USO: FUNCIÓN DONDE SE GENERA EL CONTENIDO (SDP) DE LAS PETICIONES PARA LOS MENSAJES SIP

PARÁMETROS DE ENTRADA:

ip	Dirección IP del pot utilizado
codec	Codec de voz utilizado
tcid	Identificador del teléfono (Pot)

PARÁMETROS DE SALIDA:

port	Puerto de voz utilizado 5000…..5006

VALORES DE RETORNO:

sdp	Dirección donde se copia el contenido del SDP

EVENTOS:
Ninguno

```
extern int sysClkRateGet (void);  /*Función que controla el tiempo de duración del ciclo de poleo principal del sistema (Run state)*/
```

```
*****************************************************************************
ID's DE COLAS Y TAREAS PRINCIPALES
*****************************************************************************
extern MX_TaskId  KbtelSpuTaskId;
extern MX_QueueId KbtelSpuQueueId;
extern MX_TaskId  Kbtelamu_task_id;          /* AMU task id       */
extern MX_QueueId Kbtelamu_msg_qid;                  /* Holds queue ids  */
extern MX_TaskId  KbtelNetSigTaskId;
extern MX_QueueId  KbtelNetSigQueueId;
```

Principalmente estas tareas y colas se ponen en marcha cuando el sistema se inicia por primera vez, levantando los servicios en AMU Y SPU, así como los servicios de Red.

```
*****************************************************************************
Estados por TCID
*****************************************************************************
#define kb_idle_st          0
#define kb_connect_st       1
#define kb_first_ring_st    2
#define kb_ringing_st       3
#define kb_trying_st        4
```

Estas opciones son los posibles estados del lazo principal de cada uno de los teléfonos, es decir cuando se trate de determinar si el teléfono se encuentra colgado o descolgado y a continuación se determine si el teléfono está en llamada en espera, conectado a una llamada, en ringing o está tratando de establecer la llamada.

```
extern unsigned short KBTEL_CREATE_RTP(unsigned char POT,unsigned long RTP_PEER_ADDRESS,unsigned short RTP_PEER_PORT);
```

USO: FUNCIÓN POR MEDIO DE LA CUAL SE CREAN LOS PAQUETES DEL PROTOCOLO RTP PARA LA TRANSMISIÓN DE VOZ, SIN ESTA FUNCIÓN NO SE PODRÁ ENVIAR LA VOZ AL TELÉFONO DESTINO

PARÁMETROS DE ENTRADA:

POT	Identificador del teléfono (tcid)
RTP_PEER_ADDRESS	Dirección IP destino (Servidor SIP)
RTP_PEER_PORT	Puerto de voz destino

PARÁMETROS DE SALIDA:

Ninguno

VALORES DE RETORNO:

0	Validación de acción correcta

EVENTOS:

Ninguno

extern unsigned short KBTEL_DESTROY_RTP(unsigned char POT);

USO: FUNCIÓN POR MEDIO DE LA CUAL SE ELIMINA LA CONEXIÓN DE VOZ CON UN DETERMINADO PUERTO DESTINO, LIBERÁNDOSE EL PUERTO.

PARÁMETROS DE ENTRADA:

POT	Identificador del teléfono (tcid)

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:

0	Validación de acción correcta

EVENTOS:
Ninguno

****
TCP DRIVER
****

extern unsigned short RDRV_NEW(unsigned short Idx, int argc, char *argv]);

extern unsigned short SOCK_SendTo (SOCKET LSock,char *Buffer,unsigned short LenBuffer,unsigned short RemotePort,char *RemoteIPAddress);

USO: FUNCIÓN POR MEDIO DE LA CUAL SE ENVÍA UNA CADENA DE DATOS A TRAVÉS DEL SOCKET.

PARÁMETROS DE ENTRADA:

LSock	Nombre del Socket creado
Buffer	Cadena de caracteres a enviar
LenBuffer	Tamaño del Buffer
RemotePort	Puerto SIP 5060
RemoteIPAddress	Dirección IP destino

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:

SOCK_SUCCESS	Datos enviados
1	Error al mandar los datos

EVENTOS:

Sendto	Envía datos al destino especificado

extern unsigned short SOCK_CreateTCP(SOCKET *NewSocket, unsigned short MyPort, unsigned short TCP_TYPE);

USO: FUNCIÓN CON LA QUE SE CREA UN SOCKET DE TIPO TCP (Orientado a conexión).

PARÁMETROS DE ENTRADA:

NewSocket	Nombre del Socket creado
MyPort	Puerto asignado al Socket para enviar o recibir.

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:

CreateSocket	Crea el socket indicado TCP
FFFF	Error al crear socket

EVENTOS:

CreateSocket	Crea el socket TCP

unsigned short SOCK_CreateUDP(SOCKET *NewSocket, unsigned short MyPort)

USO: FUNCIÓN CON LA QUE SE CREA UN SOCKET DE TIPO UDP (No orientado a conexión).

PARÁMETROS DE ENTRADA:

NewSocket	Nombre del Socket creado
MyPort	Puerto asignado al Socket para enviar o recibir. 5060 para SIP.

PARÁMETROS DE SALIDA:
Ninguno

VALORES DE RETORNO:

CreateSocket	Crea el socket indicado UDP

EVENTOS:

CreateSocket	Crea el socket UDP

extern unsigned short SOCK_Recv (SOCKET LSock , char *Buffer, unsigned short LenBuffer, unsigned short *LenRx);

USO: FUNCIÓN CON LA QUE SE LEEN LOS DATOS QUE LLEGAN AL PUERTO ESPECÍFICO DEL SOCKET.

PARÁMETROS DE ENTRADA:

LSock	Nombre del Socket creado
Buffer	Dirección de memoria donde se guardarán los datos que reciba por el puerto.
LenBuffer	Tamaño del Buffer.

PARÁMETROS DE SALIDA:

LenRx	Número de bytes recibidos.

VALORES DE RETORNO:

SOCK_SUCCESS	Datos recuperados satisfactoriamente.
-1	Error al recibir los datos.

EVENTOS:

PollSocket	Checa si recibió datos en el socket.

```
extern unsigned short SOCK_Close( SOCKET LSock );
```

USO: FUNCIÓN CON LA SE CIERRA UN SOCKET DE CUALQUIER TIPO, LIBERÁNDOSE ASÍ LOS RECURSOS UTILIZADOS POR ÉSTE.

PARÁMETROS DE ENTRADA:

LSock	Nombre del Socket creado

PARÁMETROS DE SALIDA:

Ninguno

VALORES DE RETORNO:

SOCK_SUCCESS	Socket cerrado satisfactoriamente.
-1	Error al cerrar el socket.

EVENTOS:

Close	Cerrar Socket.

```
#endif __KBTEL_H__
```

4.4.3.2 kbtelsh.c

El programa *kbtelsh.c* puede ser considerado como el principal de todos los que conforman la carpeta *root* con la cual se crea la imagen a descargar, ya que dentro de todas sus funciones está la de inicializar las tareas de los módulos AMU, SPU, TCP y el de red. Si las tareas en cada uno de los módulos no se inicializan, entonces no tendremos posibilidad de trabajar con procesos en paralelo o con procesos dada una cierta prioridad, puesto que éstos no estarán listos al momento de que sean requeridos.

Cada vez que la tarjeta se inicializa, debe pasar por esta parte del código para iniciar los lazos principales en cada módulo y hacer las configuraciones que a continuación se describirán, por lo que con justificada razón, *se considera el main de todo el proyecto.*

Al mismo tiempo en que *kbtelsh.c* se inicia, se debe esperar a que ggroot.c realice sus tareas, se prefiere que *ggroot.c* se configure primero antes de seguir el flujo del programa en *kbtelsh.c*

```
#include <kbtel.h>
```

Recordemos que en *kbtel.h* se encuentran los prototipos de la mayoría de las funciones, las librerías importantes, variables y constantes, por supuesto todas globales.

```
TSG_NW_ADDR_S srcAddr, desAddr;   /*Estructuras para las direcciones local y remota*/

extern KBTEL_MCB_S   kbtel_mcb;

MX_TaskId KbtelSpuTaskId;
MX_QueueId KbtelSpuQueueId;

MX_TaskId Kbtelamu_task_id = 0;                    /* AMU task id */
MX_QueueId Kbtelamu_msg_qid = 0;                   /* Holds queue ids */

MX_TaskId KbtelNetSigTaskId;
MX_QueueId KbtelNetSigQueueId;
```

Notemos que se declaran las funciones para levantar las tareas y las colas del módulo SPU, AMU y el de red.

```
*************************************************************************************************
ESTABLECE DIRECCIÓN LOCAL (TARJETA) Y REMOTA (SERVIDOR PROXY YALTEL)
*************************************************************************************************
psrcAddr=&srcAddr;
pdesAddr=&desAddr;

net_get_ip_addr((CHAR*)& (TelogyBox_addr.u_addr.ipv4_port.addr[0]));

psrcAddr->nw_protocol=TSG_NW_PROT_UDP_IPV4_PORT;
psrcAddr->u_addr.ipv4_port.port=KBTEL_BASE_PORT;
psrcAddr->u_addr.ipv4_port.addr[0]=192;  /*Direccion y puerto de la tarjeta de voz*/
psrcAddr->u_addr.ipv4_port.addr[1]=168;
psrcAddr->u_addr.ipv4_port.addr[2]=1;
psrcAddr->u_addr.ipv4_port.addr[3]=190;
```

Se asigna a la tarjeta de voz una dirección IP estática con la cual se identificará *(192.168.1.190)* además de un puerto base para voz, en este caso *kbtel_base_port=5000*.

```
psrcAddr->u_addr.ipv4_port.padding=0;

pdesAddr->nw_protocol=TSG_NW_PROT_UDP_IPV4_PORT;
pdesAddr->u_addr.ipv4_port.port=KBTEL_BASE_PORT;
pdesAddr->u_addr.ipv4_port.addr[0]=192;  /*Direccion y puerto para la localidad remota servidor proxy
                                           YALTEL*/
pdesAddr->u_addr.ipv4_port.addr[1]=168;
pdesAddr->u_addr.ipv4_port.addr[2]=1;
pdesAddr->u_addr.ipv4_port.addr[3]=189;
pdesAddr->u_addr.ipv4_port.padding=0;
```

Se asigna al servidor proxy, con el cual se establecerá la conexión, una dirección IP estática para identificarse *(207.17.190.58)* y un puerto base para voz. El puerto puede ser dinámico debido a que el servidor enviará el puerto de voz por el cual establecerá la conexión en las respuestas a las peticiones SIP, debido a esto, el puerto destino se tomará de dichas respuestas.

```
*************************************************************************************
CREA LA TAREA Y LA COLA PARA EL AMU DE KBTEL
*************************************************************************************

memset(&KbtelAmuTaskAttr, 0, sizeof(&KbtelAmuTaskAttr));
strcpy(KbtelAmuTaskAttr.name, KBTEL_AMU_NAME);
KbtelAmuTaskAttr.priority = KBTEL_AMU_PRIORITY;
KbtelAmuTaskAttr.timeSlice= 0;
KbtelAmuTaskAttr.stackAddr= 0;
KbtelAmuTaskAttr.stackSize= KBTEL_AMU_STACK_SIZE;
KbtelAmuTaskAttr.SwapIn   = 0;
KbtelAmuTaskAttr.SwapOut  = 0;
```

```
KbtelAmuTaskAttr.SwapData = 0;

if(XtCreate(&KbtelAmuTaskAttr,(MX_TaskEntry)kbtel_amu_main,0,&Kbtelamu_task_id)==ERR_NOERR
)
{
      Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL: Kb/TEL AMU Task created...");
}
memset(&KbtelAmuQueueAttr,0,sizeof(&KbtelAmuQueueAttr));
strcpy(KbtelAmuQueueAttr.name, KBTEL_AMU_QUEUE_NAME);
KbtelAmuQueueAttr.depth = KBTEL_AMU_QUEUE_DEPTH;
KbtelAmuQueueAttr.taskId= Kbtelamu_task_id;
KbtelAmuQueueAttr.mode  = MX_SWITCH;
KbtelAmuQueueAttr.evFlag = KBTEL_AMU_QUEUE_EVENT;
if(XqCreate(&KbtelAmuQueueAttr,&Kbtelamu_msg_qid)==ERR_NOERR)
{
      Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL AMU Queue created...");
}
```

En esta parte del código se crean las tareas de AMU y se establece el enlace con el lazo principal de procesamiento de AMU en *kbtelamu.c* desde donde se estarán checando los eventos y se interrumpirá en caso de que uno de ellos requiera de los recursos necesarios para completar su tarea, posteriormente regresará al lazo principal al concluir el evento.

```
*************************************************************************************************
CREA LA TAREA PARA EL SPU DE KBTEL
*************************************************************************************************

memset(&KbtelSpuTaskAttr, 0, sizeof(&KbtelSpuTaskAttr));
strcpy(KbtelSpuTaskAttr.name, KBTEL_SPU_NAME);
KbtelSpuTaskAttr.priority = KBTEL_SPU_PRIORITY;
KbtelSpuTaskAttr.timeSlice= 0;
KbtelSpuTaskAttr.stackAddr= 0;
KbtelSpuTaskAttr.stackSize= KBTEL_SPU_STACK_SIZE;
KbtelSpuTaskAttr.SwapIn   = 0;
KbtelSpuTaskAttr.SwapOut  = 0;
KbtelSpuTaskAttr.SwapData = 0;
if(XtCreate(&KbtelSpuTaskAttr,(MX_TaskEntry)kbtel_spu_main,0,&KbtelSpuTaskId)==ERR_NOERR)
{
      Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL: Kb/TEL SPU Task created...");
}
memset(&KbtelSpuQueueAttr,0,sizeof(&KbtelSpuQueueAttr));
strcpy(KbtelSpuQueueAttr.name, KBTEL_SPU_QUEUE_NAME);
KbtelSpuQueueAttr.depth = KBTEL_SPU_QUEUE_DEPTH;
KbtelSpuQueueAttr.taskId= KbtelSpuTaskId;
KbtelSpuQueueAttr.mode  = MX_SWITCH;
KbtelSpuQueueAttr.evFlag = KBTEL_SPU_QUEUE_EVENT;
if(XqCreate(&KbtelSpuQueueAttr,&KbtelSpuQueueId)==ERR_NOERR)
{
      Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL SPU Queue created...");
}
```

En esta parte del código se crean las tareas de SPU y se establece el enlace con el lazo principal de procesamiento de SPU en *kbtelspu.c* desde donde se estarán checando los eventos y se interrumpirá en caso de que uno de ellos requiera de los recursos necesarios para completar su tarea, posteriormente regresará al lazo principal al concluir el evento.

```
*************************************************************************************************
CREA LA QUEUE PARA EL MÓDULO DE SENALIZACIÓN DE RED (PROTOCOLO PROPIETARIO
PSIP)
*************************************************************************************************

memset(&KbtelNetSigTaskAttr, 0, sizeof(&KbtelNetSigTaskAttr));
strcpy(KbtelNetSigTaskAttr.name, KBTEL_NETSIG_NAME);
```

```
KbtelNetSigTaskAttr.priority = KBTEL_NETSIG_PRIORITY;
KbtelNetSigTaskAttr.timeSlice= 0;
KbtelNetSigTaskAttr.stackAddr= 0;
KbtelNetSigTaskAttr.stackSize= KBTEL_NETSIG_STACK_SIZE;
KbtelNetSigTaskAttr.SwapIn   = 0;
KbtelNetSigTaskAttr.SwapOut  = 0;
KbtelNetSigTaskAttr.SwapData = 0;
if(XtCreate(&KbtelNetSigTaskAttr,(MX_TaskEntry)kbtel_netsig_main,0,&KbtelNetSigTaskId)==ERR_NO
ERR)
{
      Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL: Kb/TEL Network Signaling Task created...");
}
memset(&KbtelNetSigQueueAttr,0,sizeof(&KbtelNetSigQueueAttr));
strcpy(KbtelNetSigQueueAttr.name, KBTEL_NETSIG_QUEUE_NAME);
KbtelNetSigQueueAttr.depth = KBTEL_NETSIG_QUEUE_DEPTH;
KbtelNetSigQueueAttr.taskId= KbtelNetSigTaskId;
KbtelNetSigQueueAttr.mode  = MX_SWITCH;
KbtelNetSigQueueAttr.evFlag = KBTEL_NETSIG_QUEUE_EVENT;
if(XqCreate(&KbtelNetSigQueueAttr,&KbtelNetSigQueueId)==ERR_NOERR)
{
      Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL Network Signaling Queue created...");
}
```

En esta parte del código se crean las tareas de protocolo propietario PSIP y se establece el enlace con el lazo principal de procesamiento de PSIP en *kbtelpp.c* desde donde se estarán checando los eventos y se interrumpirá en caso de que uno de ellos requiera de los recursos necesarios para completar su tarea, posteriormente regresará al lazo principal al concluir el evento.

```
*********************************************************************************************
CREA LA QUEUE PARA EL MODULO DE SENALIZACION DE RED (PROTOCOLO PROPIETARIO)
*********************************************************************************************

if(RDRV_NEW(0,0,0)==SUCCESS)
{
      Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL TCP Driver created...");
}
return ERR_NOERR;
```

En esta parte del código se crean las tareas de TCP además del manejador de eventos, se establece el enlace con el lazo principal de procesamiento de TCP y de creación de tareas en *kbteltcp.c* desde donde se estarán checando los eventos y se interrumpirá en caso de que uno de ellos requiera de los recursos necesarios para completar su tarea, posteriormente regresará al lazo principal al concluir el evento. El manejador de eventos TCP será el más importante, pues aquí se encontrará el lazo principal de todo el sistema y como ya se mencionó será interrumpido debido a eventos en cualquiera de los módulos.

4.4.3.3 kbteltcp.c (MÓDULO TCP)

Una vez que se ha iniciado la configuración de la tarjeta en *kbtelsh.c*, se abra puesto en marcha las tareas correspondientes a cada uno de los módulos, estas tareas son de gran importancia ya que conducirán al sistema en general a un estado de espera. Este estado de espera, mantendrá al sistema siempre alerta de los eventos que surjan como peticiones de un proceso en general, es decir, una tarea específica del sistema podrá requerir de funciones en cada módulo para completarse, por lo tanto, se realizarán los eventos necesarios en cada módulo para cumplirla.

A medida que se van levantando las tareas, el código en *kbtelsh.c* lleva a cada módulo a un lazo principal y posteriormente a un manejador de tareas; veamos lo que ocurre en el caso del módulo TCP en *kbteltcp.c*

```
*********************************************************************************************
CREA LA QUEUE PARA EL MÓDULO DE SENALIZACIÓN DE RED TCP
*********************************************************************************************

if(RDRV_NEW(0,0,0)==SUCCESS)
{
      Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL TCP Driver created...");
}
return ERR_NOERR;
```

RDRV_NEW(0,0,0) es una función de *kbteltcp.c* en donde se crean las tareas para el driver de TCP, si la respuesta es exitosa (success), el driver se abrá creado satisfactoriamente.

```
unsigned short RDRV_NEW(unsigned short Idx, int argc, char *argv])
{
      /* variables locales */
      MX_TaskAttr KbtelTcpTaskAttr;
      MX_QueueAttr KbtelTcpQueueAttr;
      unsigned short ret=0;

      /* inicia codigo */
      if (RDRV_FIRST_TIME==TRUE)
      {
            for (ret=0; ret<RDRV_MAX_INSTANCE; ret++)
                  RDRV[ret]=NULL;
            RDRV_FIRST_TIME=FALSE;
      }
      if (Idx>=RDRV_MAX_INSTANCE)
      {
            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Instance Error!!" );
            return RDRV_ERROR_INSTANCE;
      }
      if (RDRV[Idx]!=NULL)
      {
            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Already running!!");
            return RDRV_ERROR_ISRUNNING;
      }
      RDRV[Idx]=malloc(sizeof(RDRV_TYPE_CTRL));
      if (RDRV==NULL)
      {
            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Allocation memory error!!" );
            return RDRV_ERROR_ALLOCATION;
      }
      RDRV[Idx]->CTRL.NextEdo =                    99;
      RDRV[Idx]->CTRL.Lock=                        FALSE;
      RDRV[Idx]->TCP_CTRL.Edo_Rx=                  0;
      RDRV[Idx]->TCP_CTRL.LonBuffer=               0;
      RDRV[Idx]->TCP_CTRL.LonRx=                   0;
      RDRV[Idx]->TCP_CTRL.lrc=                     0;
      RDRV[Idx]->TCP_CTRL.Offset=                  0;
      RDRV[Idx]->RGW_CTRL.CountEntry=              0;
      if (GET_TCP_CONFIG(Idx,RDRV[Idx],argc,argv)!=SUCCESS)
      {
            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Configuration file error!!" );
            return RDRV_ERROR;
      }
      memset(&RDRV[Idx]->TCP_STATUS,0,sizeof(RDRV[Idx]->TCP_STATUS));
      RDRV[Idx]->CTRL.CountKeepToLive = 0;
      RDRV[Idx]->CTRL.CountResponse   = 0;
      RDRV[Idx]->CTRL.go_Thread    = TRUE;
```

```
*****************************************************************************************
CREA LA TAREA PARA EL TCP DRIVER DE KBTEL
*****************************************************************************************

// Dentro de esta función se llama a la función RDRV_EXECUTE en donde se creará  el lazo principal
de ejecución.

      /*Creando Tareas y Colas*/

      memset(&KbtelTcpTaskAttr, 0, sizeof(&KbtelTcpTaskAttr));
      strcpy(KbtelTcpTaskAttr.name, KBTEL_TCP_NAME);
      KbtelTcpTaskAttr.priority = KBTEL_TCP_PRIORITY;
      KbtelTcpTaskAttr.timeSlice= 0;
      KbtelTcpTaskAttr.stackAddr= 0;
      KbtelTcpTaskAttr.stackSize= KBTEL_TCP_STACK_SIZE;
      KbtelTcpTaskAttr.SwapIn   = 0;
      KbtelTcpTaskAttr.SwapOut  = 0;
      KbtelTcpTaskAttr.SwapData = 0;

      /*    LLAMADA A RDRV_Execute  */

      if(XtCreate(&KbtelTcpTaskAttr,(MX_TaskEntry)RDRV_Execute,0,&KbtelTcpTaskId)==ERR_NOE
RR)
      {
            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL: Kb/TEL TCP Task created...");
      }
      memset(&KbtelTcpQueueAttr,0,sizeof(&KbtelTcpQueueAttr));
      strcpy(KbtelTcpQueueAttr.name, KBTEL_TCP_QUEUE_NAME);
      KbtelTcpQueueAttr.depth = KBTEL_TCP_QUEUE_DEPTH;
      KbtelTcpQueueAttr.taskId= KbtelTcpTaskId;
      KbtelTcpQueueAttr.mode  = MX_SWITCH;
      KbtelTcpQueueAttr.evFlag = KBTEL_TCP_QUEUE_EVENT;
      if(XqCreate(&KbtelTcpQueueAttr,&KbtelTcpQueueId)==ERR_NOERR)
      {
            Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL: Kb/TEL TCP Queue created...");
      }
```

Una vez creadas las tareas y las colas de información se indicará el estado siguiente al que pasará la ejecución, una vez dentro del lazo principal, este estado siguiente dependerá de qué necesita el sistema configurar o que paso necesita dar para completar las tareas que le puedan ser asignadas dentro del módulo TCP.

```
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_KEEP_TO_LIVE_EVENT,NULL,NULL,&kbtel_mcb.Ti
merKeepToLive_id));
      MX_CHECK(XtmrStart(kbtel_mcb.TimerKeepToLive_id, sysClkRateGet() * KEEP_TO_LIVE,0));
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_RESPONSE_EVENT,NULL,NULL,&kbtel_mcb.Timer
Response_id));
      MX_CHECK(XtmrStart(kbtel_mcb.TimerResponse_id,sysClkRateGet() * RESPONSE ,0));
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_FLASH_EVENT,NULL,NULL,&TimerFlash_id[0]));
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_FLASH_EVENT,NULL,NULL,&TimerFlash_id[1]));
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_FLASH_EVENT,NULL,NULL,&TimerFlash_id[2]));
      MX_CHECK(XtmrCreate(KBTEL_TCP_TIMER_FLASH_EVENT,NULL,NULL,&TimerFlash_id[3]));
      memset(&NT_VALUES,0,sizeof(RDRV_TYPE_TCP_CFG ));
      RDRV[Idx]->CTRL.NextEdo = WAIT_CONFIRM;

      /* Edo. Siguiente WAIT_CONFIRM */

      kbtel_tcp_gen_msg(WAIT_CONFIRM,0,0,0,0,0);
      return RDRV_SUCCESS;
}
```

El estado al que deberá pasar el sistema cuando se encuentre dentro del lazo principal será: WAIT_CONFIRM.

La función RDRV_Execute crea el lazo principal de tareas desde donde se esperará responder a los eventos que ocurran y que necesiten de las funciones del

módulo TCP. El lazo principal se creará en un manejador de eventos TCP (*kbtel_tcp_event_handler*) desde donde se realizarán los procesos paralelos o jerarquizados para completar la petición. El lazo principal se maneja por medio de estados siguientes, es decir, una vez inicializado el estado con el que entrará al lazo principal (WAIT_CONFIRM) se deberá indicar cual será el siguiente estado, pues de lo contrario se creará un lazo solo dentro de este último estado.

```
static void RDRV_Execute( void *arg )
{
	MX_EventFlagSet Kbtel_Tcp_events;

	/* Main Task Loop */
	while (TRUE)
	{
		MX_CHECK(XevWait(MX_ALL_EVENTS,MX_OR_COND,MX_INDEFINITE,&Kbtel_Tcp_eve
nts));
		kbtel_tcp_event_handler(Kbtel_Tcp_events);	/*manejador de eventos TCP*/
	}
}
```

Como ya se mencionó en *kbtel_tcp_event_handler* se manejarán estados siguientes para enlazar a cada una de las etapas de configuración y de ejecución del lazo principal. Los estados son: CONNECT_TCP, SEND_ID, WAIT_CONFIRM, WAIT_CONFIG y RUN. La primera parte corresponde al estado WAIT_CONFIRM:

```
*************************************************************************************************
 FUNCTION PURPOSE:
*************************************************************************************************
DESCRIPTION: Manejador de eventos TCP
*************************************************************************************************

void kbtel_tcp_event_handler(MX_EventFlagSet tcp_events_pending)
{
	int port;
	MX_Result ret_code;
	KBTEL_MSG_T * p_msg;
	BOOL free_buffer;
	TCID tcid;
	DATA32		p1,p2,p3,p4;

	/* variables locales */

	const unsigned short Idx = 0; /*(unsigned)arg;*/
	unsigned short LonRx, ret;
	unsigned long RGW_ID;
	char TCP_BUFFER[MAX_TCP_BUFFER+1];
	static int indicador;  */
	port=5060;   /*Puerto SIP*/

	if (tcp_events_pending == KBTEL_TCP_QUEUE_EVENT)
	{
		/* Handle Inter-unit Message Queue */
		ret_code = XqWait(KbtelTcpQueueId, MX_NO_BLOCK, (MX_MessagePointer*) &p_msg);
		if (ret_code != ERR_QEMPTY)
		{
			MX_CHECK(ret_code);
			tcid = p_msg->header.tcid;
			p1   = p_msg->body.gen.param1;
			p2   = p_msg->body.gen.param2;
			p3   = p_msg->body.gen.param3;
			p4   = p_msg->body.gen.param4;
			free_buffer = TRUE;

			switch (RDRV[Idx]->CTRL.NextEdo)
			{
```

```
                    case CONNECT_TCP:
                        Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: CONNECT_TCP STATE");
                        if (ConnectToServer(Idx)==SOCK_SUCCESS)
                        {
                            RDRV[Idx]->CTRL.NextEdo = SEND_ID;
                            RDRV[Idx]->CTRL.Lock        = FALSE;
                            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Connect to Server
Success");
                            kbtel_tcp_gen_msg(SEND_ID,tcid,0,0,0,0);
                        }
                        else
                        {
                            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Connect to Server No
Success");
                            RDRV[Idx]->TCP_STATUS.Retries++;
                            kbtel_tcp_gen_msg(CONNECT_TCP,tcid,0,0,0,0);
                            XtSleep(NULL, RDRV[Idx]->TCP_CFG.TO_Reconnect *sysClkRateGet());
                        }
                    break;
                    case SEND_ID:
                        Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: SEND_ID STATE");
                        ret=RDRV_SEND_RES_ID(Idx);
                        if (ret)
                        {
                            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Fail to send Rgw Id");
                            RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                            kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                        }
                        else
                        {
                            ResetKeepToLive(Idx);
                            RDRV[Idx]->CTRL.NextEdo=WAIT_CONFIG;
                            kbtel_tcp_gen_msg(WAIT_CONFIG,tcid,0,0,0,0);
                        }
                    break;
                    case WAIT_CONFIRM:
                        if (SOCK_CreateUDP(&SipSocket,port)==SOCK_SUCCESS)
                        {
                            RDRV[Idx]->TCP_CTRL.PeerSocket=SipSocket;
                            Xspy( SPY_KBTEL, SPY_EVENT, "SipSocket se abrió con éxito,puerto
5060");
                        }
                        else
                            Xspy( SPY_KBTEL, SPY_EVENT, "SipSocket no pudo abrirse");
                        RDRV_CONFIG_POTS(0,"",0);
                        RDRV[Idx]->CTRL.NextEdo=RUN;
                        kbtel_tcp_gen_msg(RUN,1,0,0,0,0);   /*Siguiente estado RUN*/
                        indicador=0;                /*tcid*/
                        FLAG=0;
                    break;
```

Dentro del estado *WAIT_CONFIRM* se crea el socket UDP (SipSocket) para escuchar y enviar los mensajes SIP entre la tarjeta de voz y el servidor SIP, el puerto utilizado por el socket es el 5060 (port), el número de este puerto parte de una norma internacional RFC.

Una vez abierto el socket, se iniciará la configuración de cada uno de los puertos de conexión en los 6 teléfonos disponibles, *RDRV_CONFIG_POTS* configurará la dirección IP local de la tarjeta, además de asignar a cada uno de los POTS un puerto de voz por el cual se comunicará con el servidor SIP recibiendo o enviando paquetes de voz, estos puertos parten del 5000 para el pot 0, 5002 para el pot 1 y así sucesivamente. Cada uno de los teléfonos tendrá la misma dirección IP origen, lo que los distinguirá a nivel IP, será el puerto local de cada uno de ellos.

```
*************************************************************************************************
FUNCTION PURPOSE:
*************************************************************************************************
DESCRIPTION:  Configura los teléfonos POTS
*************************************************************************************************

static unsigned short RDRV_CONFIG_POTS(unsigned short Idx, char *Buffer, unsigned short LonRx)
{
      unsigned short i, offset,ret=0;
      long RTP_Local_IpAddress;
      char *RTP_Local_Ip;
      RTP_Local_Ip=&RTP_Local_IpAddress;
      RTP_Local_Ip[0]=192;    /*Ip tarjeta del gateway SIP 192.168.1.190*/
      RTP_Local_Ip[1]=168;
      RTP_Local_Ip[2]=1;
      RTP_Local_Ip[3]=190;
      RDRV[Idx]->RGW_CTRL.CountEntry=6;

            /*Asignación de puertos locales KBTEL_BASE_PORT (5000) al 5010 */

      for (i=0;i<RDRV[Idx]->RGW_CTRL.CountEntry;i++)
      {
            RDRV[Idx]->RGW_CTRL.MapEntry[i].Board=0;
            RDRV[Idx]->RGW_CTRL.MapEntry[i].Pot=i;
            RDRV[Idx]->RGW_CTRL.MapEntry[i].RTP_Local_Port=KBTEL_BASE_PORT + (i*2);

      }
      for(i=0;i<=MAX_POTS;i++)
       /*Inicializa Índices que servirán como contadores para capturar el número telefónico marcado*/
      {
            telefono[i].indice=1;
      }
      for(i=0;i<=MAX_POTS;i++)
            telefono[i].ind=0;
      Xspy(SPY_KBTEL, SPY_FNENTER,"KBTEL-tcp: Dentro de CONFIG_POTS");
      return RDRV_CHECK_POTS(Idx);
}
```

RDRV_CHECK_POTS es una función con la que se verifican cada uno de los pots configurados en *RDRV_CONFIG_POTS*, en cuanto a los puertos locales asignados a cada uno de ellos.

El estado al que pasará una vez hecho la configuración es RUN, iniciando con el indicador 0 o tcid 0. Este indicador cambiará en cada ciclo del lazo principal del sistema e irá de 0 a 5, con el cambio del indicador se podrá checar a cada pot o tcid constantemente.

```
                  case WAIT_CONFIG:
                        Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: WAIT_CONFIG State");
                        ret = RecvFromServer(Idx, TCP_BUFFER, MAX_TCP_BUFFER, &LonRx);
                        if (ret==RDRV_ERROR)
                        {
                              RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                              kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                              Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: To SHUTDOWN State");
                        }
                        else if(ret==RDRV_SUCCESS)
                        {
                              Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: SUCCESS");
                              if(TCP_BUFFER[DRV_POS_TYPE]==DRV_CTRL&&TCP_BUFFER[DRV
_POS_CMD]==DRV_RES_CFG)
                              {
                                    if (RDRV_CONFIG_POTS(Idx,TCP_BUFFER,LonRx)==
RDRV_SUCCESS)
                                    {
```

```
                                        RDRV[Idx]->CTRL.NextEdo=WAIT_CONFIRM;
                                        kbtel_tcp_gen_msg(WAIT_CONFIG,tcid,0,0,0,0);
                                    }
                                    else
                                    {
                                        RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                                        kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                                        Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Recv bad
frame (config data");
                                    }
                                }
                                else if
(TCP_BUFFER[DRV_POS_TYPE]==DRV_CTRL&&TCP_BUFFER[DRV_POS_CMD] ==
DRV_CTRL_ACK)
                                {
                                    /*TIMER_Start(&TimerTask);*/
                                    RDRV[Idx]->CTRL.NextEdo=RUN;
                                    Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Connection
Complete!");
                                    kbtel_tcp_gen_msg(RUN,tcid,0,0,0,0);
                                }
                                else
                                {
                                    RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                                    kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                                    Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: Recv Bad frame
(Ack Config)");
                                }
                            }
                            else if(ret==RDRV_EMPTY)
                            {
                                Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: RDRV_EMPTY");
                                ret=GetReceiveStatus(Idx);
                                if (ret==RDRV_TIMEOUT)
                                {
                                    RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                                    kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                                    break;
                                }
                                else if (ret==RDRV_ERROR)
                                {
                                    RDRV[Idx]->CTRL.NextEdo=SHUTDOWN;
                                    kbtel_tcp_gen_msg(SHUTDOWN,tcid,0,0,0,0);
                                    break;
                                }
                                else
                                    kbtel_tcp_gen_msg(WAIT_CONFIG,tcid,0,0,0,0);
                            }
                        break;
                        case RUN:
                            Xspy( SPY_KBTEL, SPY_EVENT, "KBTEL-tcp: RUN State");
```

Checamos timer dial, por pot, solamente si no se ha presionado ningún dígito. El timer dial se inicia por cada teléfono cuando descolgamos alguno de éstos, a la par con la generación del tono de invitación a marcar.

```
                        if(telefono[indicador].indice==1)
                        {
                            MX_CHECK(XtmrInquiry(TimerDial[indicador],&TimerDialst[indicador]));
                            if(TimerDialst[indicador]==TIMER_ACTIVE)
                                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: TIMER DIAL
ACTIVO");
```

Si el timer dial, por teléfono, se ha vencido; entonces se suspende el tono de invitación a marcar y se genera el tono de ocupado, enviando al usuario un mensaje de cuelgue por favor.

```
                        if(TimerDialst[indicador]==TIMER_FIRED)
                        {
                            spup_tone_gen(indicador,TSG_TONE_BUSY,FALSE,TRUE);
                            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: CUELGUE POR
FAVOR");
                        }
                    }
```

Si se ha presionado un dígito, el tono de invitación a marcar se suspende al igual que el timer dial, entonces, se activa un timer de marcación que corresponderá al tiempo total que tiene el usuario para marcar el número telefónico.

```
                    if(telefono[indicador].indice!=1 && FLAG==0)
                    {
                        MX_CHECK(XtmrInquiry(TimerPots[indicador],&TimerPotst[indicador]));
                        /*Checamos timer número marcado*/
                        if(TimerPotst[indicador]==TIMER_ACTIVE)
                            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: TIMER
MARCACION CONT ACTIVO");
```

Una vez que se ha vencido el tiempo para marcar el número teléfonico, *telefono[indicador].string2* almacenará el número marcado y se creará la peticiòn de inicio de sesión SIP INVITE.

```
                        if(TimerPotst[indicador]==TIMER_FIRED)
                        {
                            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu:
TELEFONO:%s",telefono[indicador].string2);
                            create_request(indicador,"INVITE",create_sdp(config.ipsalida,5002,
8,indicador),"");
                            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: IP
LOCAL:%s***",config.ip_local);
                            FLAG=1;
                        }
                    }
                    EXEC_ADM_TASK();
                    indicador=(indicador + 1)%6;  /*Contador de 0 a 5 que corresponde al tcid y
cambia en cada ciclo RUN */
```

En cada ciclo RUN se checa si el socket ha recibido datos a través de la función *RecvFromServer*.

```
                    ret = RecvFromServer(Idx,TCP_BUFFER, MAX_TCP_BUFFER, &LonRx);
                    kbtel_tcp_gen_msg(RUN,0,0,0,0,0);  /*Estado siguiente RUN*/
                break;
            }
```

Siempre que entremos al estado de RUN, se realizará el mismo procedimiento creando un ciclo infinito. Si en algún otro módulo se realiza una tarea, se interrumpirá el ciclo para cumplir la tarea y continuará el ciclo al finalizar.

En si, el ciclo infinito de RUN consistirá en checar los timers de los tonos generados o los timers de marcación de dígitos, para poder iniciar la sesión SIP. Además, se checará en cada ciclo si el socket ha recibido algún frame de datos y así poder enviar una respuesta a mensajes SIP.

```
                Xspy( SPY_KBTEL, SPY_GEN_INFO, "KBTEL-tcp: TCP-DRV State %d", RDRV[Idx]->CTRL.NextEdo);
                if(free_buffer)
                    kbtel_tcp_free_msg(p_msg);
                XtSleep(NULL, sysClkRateGet() * 0.5);
            }
        }
}
```

En la siguiente sección del código se describirán las funciones que capturan frames de mensajes que son recibidos en el socket UDP:

```
*************************************************************************************************
FUNCTION PURPOSE:
*************************************************************************************************
DESCRIPTION:  Recibe datos por el socket UDP
*************************************************************************************************
/* Regresa result:
        ERROR: El socket falló o se perdió la conexión.
        SUCCESS: Se recibió bien el frame.
        EMPTY: El socket no recibió nada.    */

static unsigned short RecvFromServer(unsigned short Idx, char *buffer, unsigned short SizeBuffer,
unsigned short *LonRx)
{

    unsigned short          result;
    unsigned long Addr;
    unsigned long RGW_ID;
    unsigned int index;
    Addr=RDRV[Idx]->RGW_CTRL.RGW_ID;
    result=ReceiveFrame(Idx,buffer, SizeBuffer, LonRx);
    switch (result)
    {
        case SOCK_SUCCESS:
            result = RDRV_EMPTY;
            if (*LonRx >= DRV_HEADER_LON)
            {
                MemCpy(&RGW_ID, &buffer[DRV_POS_ID], DRV_LON_ID);
                if (RGW_ID==Addr)
                {
                    Xspy(SPY_KBTEL, SPY_GEN_INFO,"KBTEL-tcp: RGW_ID=Addr");
                    ResetKeepToLive(Idx);
                    result = RDRV_SUCCESS;
                }
                else
                {
                    Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Bad frame (invalid Rgw Id)");
                }
            }
            else
            {
                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Bad frame (incomplete)");
            }
        break;
        case SOCK_EMPTY:
            Xspy(SPY_KBTEL, SPY_GEN_INFO,"KBTEL-tcp: SOCK_EMPTY");
            result = RDRV_EMPTY;
        break;
        default:
            Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Socket fail or connection reset by peer");
            result = RDRV_ERROR;
        break;
    }
    return result;
}
```

```
*************************************************************************************************
FUNCTION PURPOSE:
*************************************************************************************************
DESCRIPTION:  Identifica y recibe los datos socket UDP
*************************************************************************************************

static unsigned short ReceiveFrame(unsigned short Idx, char *BufferOut, unsigned short SizeBuffer,
unsigned short *LonOut)
{

      unsigned short result;
      int contador;
      char xByte;
      char *msgIN, *msgAudio, *msgTag;
      TCID tcid;
      RDRV_TYPE_TCP_CTRL *ptr = NULL;
      *LonOut=0;
      ptr = &RDRV[Idx]->TCP_CTRL;
      memset(ptr->BufferRx,0,sizeof(ptr->BufferRx));          // Limpia el buffer ptr->BufferRx guardará lo
que reciba el socket

      Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Receive Frame Function %d ", ptr->LonBuffer);
      if (ptr->LonBuffer==0)
      {
            ptr->Offset=0;
```

SOCK_Recv trae los datos recibidos en caso de existir.

```
            result=SOCK_Recv(ptr->PeerSocket, ptr->BufferRx, SizeBuffer, &ptr->LonBuffer);
            Xspy(SPY_KBTEL, SPY_FNENTER,"KBTEL-tcp: result 0x%x", result);
```

Si entramos al siguiente if entonces hubo error en la recepción.

```
            if (result!=SOCK_SUCCESS && result!=SOCK_EMPTY)
            {
                  Xspy(SPY_KBTEL,SPY_EVENT,"KBTEL-tcp:Sock_Recv !=SOCK_SUCCESS or
!=SOCK_EMPTY");
                  return result;
            }
```

Si ptr->LonBuffer tiene valor 1 entonces result= SOCK_SUCCESS y ptr->BufferRx contendrá el frame recibido.

```
            if (ptr->LonBuffer)
            {
                  Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: SOCK_Recv = SOCK_SUCCESS");
                  msgTag=strstr(ptr->BufferRx,"tag=62340930");
                  tcid=msgTag[12]-48;
```

Buscamos la cadena tag=62340930x en ptr->BufferRx, el valor de x corresponderá al tcid (identificador de teléfono) al que se le debe enviar el mensaje recibido en el socket, ya sea de SIP o de voz.

```
                  Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: TCID=%d",tcid);
```

A continuación se revisa a que respuesta sip corresponde el frame recibido: TRYING, RINGING Y OK.

```
                  msgIN=strstr(ptr->BufferRx,"Trying");
                  if (msgIN!=NULL)
                  {
                        mensaje[tcid].actual="Trying";
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Trying RECIBIDO");
```

```
                }
                else
                        msgIN=strstr(ptr->BufferRx,"Ringing");
                if (msgIN!=NULL)
                {
                        mensaje[tcid].actual="Ringing";
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Ringing RECIBIDO");
                }
```

La respuesta OK es muy importante ya que de aquí se tomará el valor del puerto de voz del servidor SIP.

```
                else
                        msgIN=strstr(ptr->BufferRx,"OK");
                if (msgIN!=NULL)
                {
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: OK RECIBIDO");
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: IP_SERVER=%s",config.ip_server);
                        mensaje[tcid].actual="OK";
                        msgAudio=strstr(msgIN,"audio");
                           if (msgAudio!=NULL)
                           {
                                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: AUDIO ENCONTRADO");
                                for(contador=0;contador<=4;contador++)
                                     header[tcid].port[contador]=msgAudio[6+contador];
```

header[tcid].port guardará el puerto de voz del servidor SIP.

```
                                header[tcid].port[5]=0;
                                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Port=%s",header[tcid].port);
```

Se envía un ACK al servidor SIP después de recibir el OK satisfactoriamente.

```
                                create_request(tcid,"ACK","","");
                                Xspy(SPY_KBTEL,SPY_EVENT,"peer_addr=%x,port=%s",ConvStrToLong(
config.ip_server),header[tcid].port);
```

Se crea la conexión de voz con el servidor SIP en la dirección IP y puerto indicado utilizando el protocolo RTP (*KBTEL_CREATE_RTP*).

```
                                KBTEL_CREATE_RTP((unsigned char)tcid,
ConvStrToLong(config.ip_server),atoi(header[tcid].port));
                                kbtel_mcb.kb_pre_st_on[tcid]=kb_connect_st;
                                kbtel_mcb.kb_st[tcid]=kb_connect_st;
                           }
                }
                else
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-tcp: Mensaje NO válido");
        }
    }
    if (ptr->LonBuffer > 0)
    {
        for (;ptr->LonBuffer; )
        {
            ptr->LonBuffer--;
            xByte=ptr->BufferRx[ptr->Offset++];
            switch (ptr->Edo_Rx)
            {
                case 0:                 // espera DLE
                     ptr->Edo_Rx = (xByte == DLE) ? 1: 0;
                     ptr->LonRx=0; ptr->lrc = 0;
                break;
                case 1:                 // espera STX
                     ptr->Edo_Rx = (xByte == STX) ? 2 : 0;
                break;
```

```
                        case 2:             // Espera Dato o DLE
                            if (xByte == DLE)
                                ptr->Edo_Rx = 3;
                            else
                            {
                                ptr->BufferTmp[ptr->LonRx++] = xByte;
                                ptr->lrc = ptr->lrc ^ xByte;
                            }
                        break;
                        case 3:             // espera DLE o ETX
                            if (xByte == DLE)
                            {
                                ptr->BufferTmp[ptr->LonRx++] = xByte;
                                ptr->lrc = ptr->lrc ^ xByte;
                                ptr->Edo_Rx = 2;
                            }
                            else if (xByte == ETX)
                            {
                                ptr->Edo_Rx = 0;
                                if (ptr->lrc == 0)
                                {
                                    ptr->LonRx = ptr->LonRx-1;
                                    memcpy(BufferOut,ptr->BufferTmp,ptr->LonRx);
                                    *LonOut = ptr->LonRx;
                                    return SOCK_SUCCESS;
                                }
                            }
                            else
                                ptr->Edo_Rx = 0;
                        break;
                    }
                }
            }
            return SOCK_EMPTY;
}
```

4.4.3.4 kbtelspu.h (MÓDULO SPU)

Una vez dentro del módulo TCP y en el ciclo infinito RUN, como ya se explicó anteriormente, la aplicación será capaz de formar un mensaje de petición SIP INVITE con el cual se podrá iniciar una sesión de tipo SIP, pero para lograr lo anterior será necesario dirigirse al módulo SPU.

Dentro del SPU existe un lazo principal (*kbtel_spu_main*) que es interrumpido cuando se presenta un evento, para dar paso al manejador de eventos (*kbtel_spup_event_handler*).

```
void kbtel_spu_main(void)
{
    MX_EventFlagSet     Kbtel_Spu_events;

     /*Main Task Loop */
    while (TRUE)
    {
        MX_CHECK(XevWait(MX_ALL_EVENTS,MX_OR_COND,MX_INDEFINITE,&Kbtel_Spu_eve
nts));
        kbtel_spup_event_handler(Kbtel_Spu_events);
     }
}
```

El manejador de eventos (*kbtel_spup_event_handler)* nos direccionará dentro del código al generador de mensajes (*kbtel_spu_dispatch_gen_msg*) desde donde se podrán controlar funciones importantes para cada uno de los teléfonos. Estas funciones están orientadas a los estados de conexión en los teléfonos tales como

colgado, descolgado, generación de tonos de invitación a marcar, de ocupado, además de los timers o contadores de tiempo para cada uno de los tonos y para la marcación de un número telefónico.

Además de las funciones anteriores, se realiza la verificación del código de protección antifraude al momento de marcar un número telefónico, así como también se captura cada uno de los dígitos marcados para poder almacenar el número por completo.

Los casos en donde se llevan a cabo las más importantes funciones son *case SPU_MSG_LOCAL_HOOK* y *case SPU_MSG_TELE_DIGIT*.

```
*****************************************************************************************
FUNCTION PURPOSE:
*****************************************************************************************
DESCRIPTION:  Procesa Mensajes
*****************************************************************************************

void kbtel_spu_dispatch_gen_msg(KBTEL_MSG_T* p_msg)
{
      TCID tcid;
      KBTEL_MSG_GEN_BODY_T* p_gen;
      DATA32 param1;
      DATA32 param2;
      DATA32 param3;
      DATA32 param4;
      KBTEL_TCID_STATE tcid_st;
      MX_TimerState TimerStateFlash;
      MX_TimerId TimerIdFlash;
      GG_CID_INFO_T *cid_p;
      const SPU_SIG_PROTO_IF_T*   p_sig_proto;
      char prot1[ ]={10,11,12,13}; /*Código de Protección Antiprotección abcd a=10, b=11, c=12, d=13*/
      tcid  = p_msg->header.tcid;
      p_gen = &p_msg->body.gen;
      param1= p_msg->body.gen.param1;
      param2 = p_msg->body.gen.param2;
      param3 = p_msg->body.gen.param3;
      param4 = p_msg->body.gen.param4;
      p_sig_proto = spup_mcb.tcids[tcid].p_sig_proto;
      switch (p_msg->header.call_id)
      {
            case SPU_MSG_CONFIG:
                  spup_proc_configure(tcid,(Tsg_sig_proto) p_gen->param1,(Tsg_mode) p_gen-
>param2);
            break;
            case SPU_MSG_UNCONFIG:
                  spup_proc_unconfigure(tcid);
            break;
            case SPU_MSG_SWITCH_IN:
                  spup_proc_switch_in(tcid);
            break;
            case SPU_MSG_LOCAL_HOOK:
                  Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Local hook=%d",tcid, (SINT) p_gen-
>param1);
                  tcid_st=kbtel_mcb.kb_st[tcid];

                  /*Levanta timer dial tone*/

                  MX_CHECK(XtmrCreate(0x0001,NULL,NULL,&TimerDial[tcid]));
                  if(XtmrStart(TimerDial[tcid],sysClkRateGet() * IDLE,0)== ERR_NOERR)
                        Xspy(SPY_KBTEL, SPY_EVENT,"TIMER DIAL TONE POT:%d",tcid);
                  spup_tone_gen(tcid,TSG_TONE_DIAL,FALSE,TRUE);  /*Genera dial tone*/
                  telefono[tcid].string2[0]=0;            /*Inicializa cadena que almacenará número marcado*/
```

En esta sección del código se generó el dial tone y su timer de duración debido a que se descolgó por lo menos uno de los teléfonos.

TELÉFONO DESCOLGADO

Pasamos al estado *idle_st* y posteriormente al estado *connect_st* enviando el mensaje de teléfono descolgado. **param1** nos indica si el teléfono está descolgado o no.

```
if(param1==TRUE)
{
    telefono[tcid].string2[0]=0;     /*Inicializa cadena de numero marcado*/
    MX_CHECK(XtmrInquiry(TimerFlash_id[tcid],&TimerStateFlash));
    if(TimerStateFlash==TIMER_ACTIVE)
    {
        MX_CHECK(XtmrAbort(TimerFlash_id[tcid]));
        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Flash",tcid);
        kbtel_netsig_gen_msg(KBTEL_MSG_FLASH_HOOK,tcid,0,0,0,0);
    }
    else
    {
        Xspy(SPY_KBTEL, SPY_GEN_INFO,"KBTEL-spu: %d, tcid_st=%d,
kb_st=%d",tcid, tcid_st, kbtel_mcb.kb_st[tcid]);
```

En este momento se verifica el estodo del POT

```
                switch(tcid_st)
                {
                    case kb_idle_st:
                        Xspy(SPY_KBTEL, SPY_EVENT," Nos encontramos en idle st");
                        kbtel_mcb.kb_st[tcid]=kb_connect_st;
                        kbtel_mcb.kb_pre_st_on[tcid]=kb_idle_st;
                    break;
                    case kb_connect_st:
                        Xspy(SPY_KBTEL, SPY_EVENT," Nos encontramos en connect
st");
                        kbtel_netsig_gen_msg(KBTEL_MSG_OFF_HOOK,tcid,0,0,0,0);
                        kbtel_mcb.kb_st[tcid]=kb_connect_st;
                        Xspy(SPY_KBTEL, SPY_EVENT,"en connect st");
                    break;
                    case kb_ringing_st:
                        kbtel_netsig_gen_msg(KBTEL_MSG_OFF_HOOK,tcid,0,0,0,0);
                        spup_mcb.cfg.p_smc_spu_tiu->set_ring_cad(tcid, FALSE,
TSG_RING_DEFAULT);
                        KBTEL_CREATE_RTP((unsigned char)tcid,
                        kbtel_mcb.tcid_tofrom_udp[tcid].rtp_peer_address,
                        kbtel_mcb.tcid_tofrom_udp[tcid].rtp_peer_port);
                        kbtel_mcb.kb_pre_st_on[tcid]=kb_ringing_st;
                        kbtel_mcb.kb_st[tcid]=kb_connect_st;
                    break;
                    default:
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Invalid tcid
state",tcid);
                    break;
                }
                amu_add_off_hook(tcid);  /*Update offhook statistics*/
            }
        }
```

TELÉFONO COLGADO

```
    else
    {
        if(telefono[tcid].indice==1)
            MX_CHECK(XtmrDelete(TimerDial[tcid]));  /*Detengo timer dial tone*/
        if(telefono[tcid].indice!=1)
        {
            MX_CHECK(XtmrDelete(TimerPots[tcid]));/*Detengo timer de marcación de
dígitos*/
```

```
                    }
                    MX_CHECK(XtmrDelete(TimerPotsB[tcid]));  /* Detengo Timer Busy Tone */
                    if(Flash[tcid]==0)
                    {
                        TimerIdFlash=TimerFlash_id[tcid];
                        MX_CHECK(XtmrStart(TimerIdFlash,sysClkRateGet() * FLASH ,0));
                        Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Flash Timer On",tcid);
                        telefono[tcid].indice=1;
                        telefono[tcid].ind=0;
                        FLAG=0;
                    }
                    else
                    {
                        Flash[tcid]=0;
                        switch(tcid_st)
                        {
                            case kb_connect_st:
                                if(kbtel_mcb.kb_pre_st_on[tcid]==kb_ringing_st||kbtel_mcb.kb_pre_s
t_on[tcid]==kb_first_ring_st)
                                {
                                    kbtel_netsig_gen_msg(KBTEL_MSG_ON_HOOK,tcid,0,0,0,0);
                                    kbtel_mcb.kb_st[tcid]=kb_connect_st;
                                }
                                else
                                {
```

En esta sección se han detenido los timers de cualquier tono generado ya sea: ocupado, en espera o de marcación.

Una vez que se ha colgado alguno de los teléfonos, el tcid (POT) nos indicará qué teléfono colgó y de inmediato se enviará un mensaje SIP BYE para cerrar la sesión y terminar la conexión.

```
                                    Xspy(SPY_KBTEL,SPY_EVENT,"enviandoBYE,TCID=%d",tcid
);
                                    create_request(tcid,"BYE","","BYE");
                                    kbtel_netsig_gen_msg(KBTEL_MSG_DISCONNECT,tcid,0,0,0,
0);
                                }

                            break;
                            default:
                                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Invalid tcid
state",tcid);
                            break;

                        }
                        amu_add_on_hook(tcid);  /*Update onhook statistics*/
                    }
                }
            break;
            case SPU_MSG_TELE_DIGIT:
                if(param2==FALSE)
                {
```

Si no se ha vencido el timer de invitación a marcar entonces, utilizando el case *SPU_MSG_TELE_DIGIT* el usuario podrá marcar el número telefónico y establecer la conexión, en caso de que el timer de invitación se termine y se inicie el tono de ocupado, ya no se podrá marcar el número telefónico y se enviará el mensaje de cuelgue por favor.

```
                    if(TimerDialst[tcid]!=TIMER_FIRED)
                    {
                        if(telefono[tcid].indice==1)
                        {
```

```
                                MX_CHECK(XtmrDelete(TimerDial[tcid]));

                                /*Kill dial tone y genera tono de silencio*/
                                spup_tone_gen(tcid,TSG_TONE_SILENCE,FALSE,FALSE);

                                /*Levanta timer número marcado*/
                                MX_CHECK(XtmrCreate(0x0001,NULL,NULL,&TimerPots[tcid]));

                                if(XtmrStart(TimerPots[tcid],sysClkRateGet() * IDLE,0)== ERR_NOERR)
                                    Xspy(SPY_KBTEL,SPY_EVENT,"TIMER MARCACION
POT:%d",tcid);
                            }

                            Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL-spu: %d, DTMF detected = %d,
ind= %d digito= %d ", tcid, param1,telefono[tcid].ind,telefono[tcid].indice);
                            tcid_st=kbtel_mcb.kb_st[tcid];
                            switch(tcid_st)
                            {
                                case kb_connect_st:
                                    mxp_int_conv(telefono[tcid].digitos,param1,10);  /*Dígito marcado
en telefono[tcid].digitos*/
```

A continuación se compara cada dígito marcado (*telefono[tcid].digitos*) con el código de protección antifraude abcd (*prot1[telefono[tcid].ind]*). Si el dígito no coincide se envía el mensaje de error de marcación. La cadena *telefono[tcid].string2* almacena el número telefónico completo marcado por el usuario.

```
                                if(strcmp(telefono[tcid].digitos[telefono[tcid].ind],prot1[telefono[tci
d].ind])
                                {
                                    if(telefono[tcid].ind>3) /*digitos telefonicos utiles a partir de ind
4 e indice 5*/
                                    {
                                        telefono[tcid].indice=telefono[tcid].indice + 1;
                                        telefono[tcid].ind=telefono[tcid].ind + 1;
                                        if(telefono[tcid].indice==4)
                                        {
                                            strcpy(telefono[tcid].string2,telefono[tcid].digitos);
                                            Xspy(SPY_KBTEL,SPY_EVENT,"NUMERO:%s",telef
ono[tcid].string2);
                                        }
                                        else
                                        {
                                            strcat(telefono[tcid].string2,telefono[tcid].digitos);
                                            Xspy(SPY_KBTEL,SPY_EVENT,"NUMERO:%s",telef
ono[tcid].string2);
                                        }
                                    }
                                    else
                                        Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL-spu:
%d,ERROR DE MARCACION",tcid);
                                }
                                else
                                {
                                    telefono[tcid].indice=telefono[tcid].indice + 1;
                                    telefono[tcid].ind=telefono[tcid].ind + 1;
                                }
                            break;
                            default:
                                Xspy(SPY_KBTEL, SPY_EVENT,"KBTEL-spu: %d, Invalid tcid
state",tcid);
                            break;
                        }
                    }
                    else
                        Xspy(SPY_KBTEL, SPY_EVENT, "KBTEL-spu: CUELGUE POR FAVOR");
```

```
                }
            break;
        }
}
```

La función ***create_request*** nos permite generar peticiones de mensajes SIP, como el teléfono público solo recibirá llamadas, entonces no será necesario crear una función para generar respuestas tales como: Ringing, Trying u Ok. Las peticiones de SIP podrán ser: INVITE, ACK, BYE, CANCEL.

Para el caso del INVITE, la función lo generará incluyendo el cuerpo del mensaje SDP previamente conformado. Entonces la sintaxis de la función será:

create_request(indicador,"INVITE",**create_sdp**(config.ipsalida,5002,8,indicador),"");

Inicialmente se creará el contenido SDP:

```
*****************************************************************************************
FUNCTION PURPOSE:
*****************************************************************************************
DESCRIPTION:  Crear SDP
*****************************************************************************************
char *create_sdp(unsigned char *ip, unsigned short port,unsigned short codec,unsigned short tcid)
{
    char crlf[3],portc[6],*codecc,*sdp,sdp_port[5]; /*codecc[3]*/
    strcpy(config.ip_local,"192.168.1.190");  /*IP Local Tarjeta del Gateway SIP*/
    strcpy(config.ip_server,"192.168.1.189");
    strcpy(config.cuenta,"6642315019");   /*Cuenta Yaltel*/
    strcpy(config.proxy,"207.17.190.58");   /*IP servidor SIP*/
    strcpy(config.ipsalida,"201.154.109.210");  /*IP pública local Tarjeta del Gateway SIP*/
    sdp=malloc(1000);
    sdp[0]=0;       /*Inicializa el buffer de sdp*/
    crlf[0]=0xD;
    crlf[1]=0xA;
    crlf[2]=0x0;
    mxp_int_conv(sdp_port,5000+(tcid*2),10);   /*Puertos de Voz Local Tarjeta SIP 5000 – 5010*/
    sdp_port[4]=0;
    strcpy(sdp,crlf);
    strcat(sdp,"v=0");
    strcat(sdp,crlf);
    strcat(sdp,"o=- 3319322138 3319322138 IN IP4 ");
    strcat(sdp,ip); /*config.ipsalida*/
    strcat(sdp,crlf);
    strcat(sdp,"s=-");
    strcat(sdp,crlf);
    strcat(sdp,"c=IN IP4 ");
    strcat(sdp,ip); /*config.ipsalida*/
    strcat(sdp,crlf);
    strcat(sdp,"t=0 0");
    strcat(sdp,crlf);
    strcat(sdp,"m=audio ");
    strcat(sdp,sdp_port);
    strcat(sdp," RTP/AVP ");
    strcat(sdp,"8 0");  */8= Códec G.711*/
    return(sdp);   /*Regresa una dirección de memoria cuyo contenido es el SDP*/
}
```

Es importante señalar que la dirección IP local de la tarjeta es una dirección privada dentro de una red. Al momento de enviar un mensaje a un punto distinto en otra red, necesita una IP pública distinta de la privada con la cual se identificará,

lo que hará diferente a cada teléfono conectado que utilice la misma IP pública será el puerto de voz que utilice.

En el servidor SIP, se dará de alta la IP pública que utilizará la tarjeta para establecer las sesiones, ya que de lo contrario no se dará acceso por parte del servidor a la red telefónica.

Una vez conformado el SDP, se formará el mensaje INVITE y el resto de las peticiones SIP con la función *create_request*.

```
*****************************************************************************************
FUNCTION PURPOSE:
*****************************************************************************************
DESCRIPTION:  Crear Petición
*****************************************************************************************

unsigned short create_request(TCID tcid,char *method,char *sdp,char *id)
{
	unsigned short x=0,z=0,i=0,y=0,ret=0,status,numero;
	char string1[600],crlf[3],iplocal[16],*aux,auxstr[5];
	char metodo[15];
	string1[0]=0;   /* Inicializamos el mensaje a enviar*/
	crlf[0]=0xD;
	crlf[1]=0xA;
	crlf[2]=0x0;
	aux=strstr(method,"INVITE");   /*method contiene el nombre de la petición SIP*/
	if(aux!=NULL)
		x=1;
	aux=strstr(method,"REGISTER");
	if(aux!=NULL)
		x=2;
	aux=strstr(method,"ACK");
	if(aux!=NULL)
		x=3;
	aux=strstr(method,"OPTIONS");
	if(aux!=NULL)
		x=4;
	aux=strstr(method,"BYE");
	if(aux!=NULL)
		x=5;
	aux=strstr(method,"CANCEL");
	if(aux!=NULL)
		x=6;
	aux=strstr(id,"BYE");
	if(aux!=NULL)
		z=0;
	aux=strstr(id,"OK");
	if(aux!=NULL)
		z=1;
	header[tcid].callid="8AFD42A0-2D92-40EA-A67A-26AE5D94DC41";  /*Call-Id específico*/
	strcpy(header[tcid].para,"<sip:");
	strcat(header[tcid].para,telefono[tcid].string2);          /*# telefónico destino*/
	strcpy(header[tcid].desde,"<sip:");
	strcat(header[tcid].desde,config.cuenta);  /* Cuenta de acceso al Servidor SIP */
	header[tcid].via="SIP/2.0/UDP 192.168.1.190:5060";
	header[tcid].tagpara="8942968";   /*Se asigna un tagpara*/
	switch(tcid)     // Se asigna un tagdesde dependiendo del TCID correspondiente para direccionar
respuestas SIP
	{
		case 0:
			header[tcid].tagdesde="623409300";
		break;
		case 1:
			header[tcid].tagdesde="623409301";
		break;
		case 2:
```

```
                header[tcid].tagdesde="623409302";
            break;
            case 3:
                header[tcid].tagdesde="623409303";
            break;
            case 4:
                header[tcid].tagdesde="623409304";
            break;
            case 5:
                header[tcid].tagdesde="623409305";
            break;
        }
        switch(x)
        {
            case 1:                         /* Formamos Mensaje INVITE*/
                strcpy(string1,"INVITE sip:");
                strcat(string1,telefono[tcid].string2);
                strcat(string1,"@");
                strcat(string1,config.proxy);   /*proxy*/
                strcat(string1," SIP/2.0");
                strcat(string1,crlf);
                strcat(string1,"Content-Length: 116");
                strcat(string1,crlf);
                strcat(string1,"Contact: ");
                strcat(string1,header[tcid].desde);
                strcat(string1,"@");
                strcat(string1,config.ipsalida);
                strcat(string1,":5060>");
                strcat(string1,crlf);
                strcat(string1,"Call-ID: ");
                strcat(string1,header[tcid].callid);
                strcat(string1,"@");
                strcat(string1,config.ip_local);
                strcat(string1,crlf);
                strcat(string1,"Content-Type: ");
                strcat(string1,"application/sdp");
                strcat(string1,crlf);
                strcat(string1,"Max-Forwards: 70");
                strcat(string1,crlf);
                strcat(string1,"From:\"Administrator\"");
                strcat(string1,header[tcid].desde);
                strcat(string1,"@");
                strcat(string1,config.proxy);   /*proxy*/
                strcat(string1,">;tag=");
                strcat(string1,header[tcid].tagdesde);
                strcat(string1,crlf);
                strcat(string1,"CSeq: ");
                strcat(string1,"1 INVITE");
                strcat(string1,crlf);
                strcat(string1,"To: ");
                strcat(string1,header[tcid].para);
                strcat(string1,"@");
                strcat(string1,config.proxy);
                strcat(string1,">");
                strcat(string1,crlf);
                strcat(string1,"Via: ");
                strcat(string1,header[tcid].via);
                strcat(string1,crlf);
                strcat(string1,"User-Agent: SJLabs-SJPhone/1.30.229b");
                strcat(string1,crlf);
                strcat(string1,sdp);
                strcat(string1,crlf);
                Xspy(SPY_KBTEL, SPY_EVENT,"Long INVITE en CREATE REQUEST es
%d",strlen(string1));
```

Envía el mensaje (*string1)* cuyo contenido forma un INVITE, por el puerto 5060 del socket SipSocket, a la dirección IP del servidor SIP (*config.proxy*). De la misma forma se realiza para los mensajes ACK, BYE, etc.

SOCK_SendTo es una función de envío de datos a través de un socket.

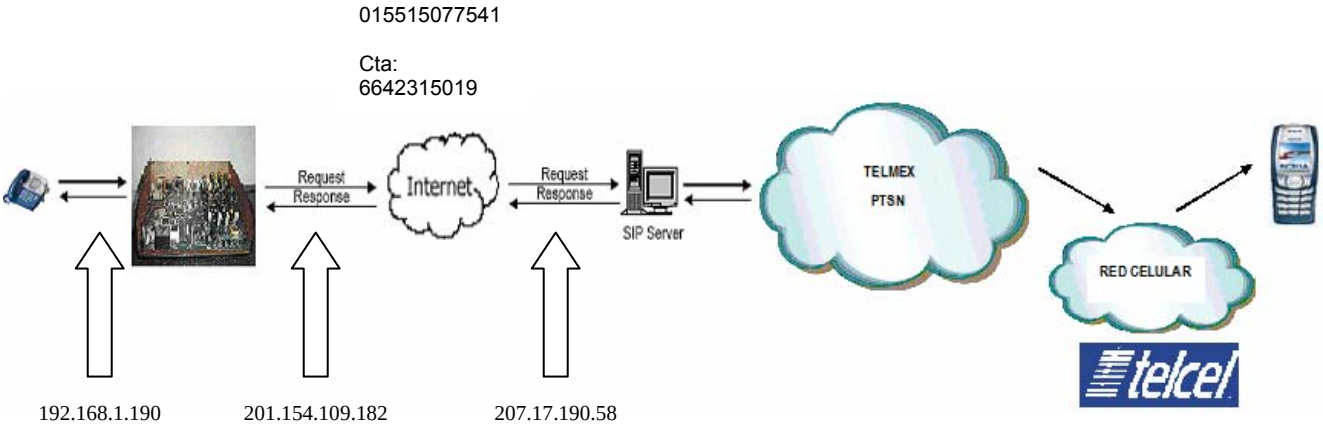
```
            if(status=SOCK_SendTo(SipSocket,string1,550,5060,config.proxy)!=SOCK_SUCCES
S)
                  Xspy(SPY_KBTEL, SPY_EVENT,"Error al mandar el INVITE");
            else
                  Xspy(SPY_KBTEL, SPY_EVENT,"INVITE Enviado");
      break;
      case 2:
      break;
      case 3:                    /*Formamos Mensaje ACK*/

            strcpy(string1,"ACK sip:");
            strcat(string1,telefono[tcid].string2);
            strcat(string1,"@");
            strcat(string1,config.proxy);
            strcat(string1,":5060");
            strcat(string1," SIP/2.0");
            strcat(string1,crlf);
            strcat(string1,"Content-Length: 0");
            strcat(string1,crlf);
            strcat(string1,"Contact: ");
            strcat(string1,header[tcid].desde);
            strcat(string1,"@");
            strcat(string1,config.ipsalida);
            strcat(string1,":5060>");
            strcat(string1,crlf);
            strcat(string1,"Call-ID: ");
            strcat(string1,header[tcid].callid);
            strcat(string1,"@");
            strcat(string1,config.ip_local);
            strcat(string1,crlf);
            strcat(string1,"Max-Forwards: 70");
            strcat(string1,crlf);
            strcat(string1,"CSeq: ");
            strcat(string1,"1");
            strcat(string1," ACK");
            strcat(string1,crlf);
            strcat(string1,"From: ");
            strcat(string1,header[tcid].desde);
            strcat(string1,"@");
            strcat(string1,config.proxy);
            strcat(string1,">;tag=");
            strcat(string1,header[tcid].tagdesde);
            strcat(string1,crlf);
            strcat(string1,"To: ");
            strcat(string1,header[tcid].para);
            strcat(string1,"@");
            strcat(string1,config.proxy);
            strcat(string1,">;tag=");
            strcat(string1,header[tcid].tagpara);
            strcat(string1,crlf);
            strcat(string1,"User-Agent: SJLabs-SJPhone/1.30.229b");
            strcat(string1,crlf);
            strcat(string1,"Via: ");
            strcat(string1,header[tcid].via);
            strcat(string1,crlf);
            Xspy(SPY_KBTEL, SPY_EVENT,"Long string1 en ACK  %d",strlen(string1));
            if(status=SOCK_SendTo(SipSocket,string1,450,5060,config.proxy)!=SOCK_SUCCES
S)
                  Xspy(SPY_KBTEL, SPY_EVENT,"Error al mandar el ACK");
            else
                  Xspy(SPY_KBTEL, SPY_EVENT,"ACK Enviado");
      break;
      case 4:
      break;
      case 5:                    /*Formamos BYE y 200 O.K.
            switch(z)
```

```
{
    case 0: /*Formamos Mensaje BYE*/
        strcpy(string1,"BYE sip:");
        strcat(string1,telefono[tcid].string2);
        strcat(string1,"@");
        strcat(string1,config.proxy);
        strcat(string1,":5060");
        strcat(string1," SIP/2.0");
        strcat(string1,crlf);
        strcat(string1,"Content-Length: 0");
        strcat(string1,crlf);
        strcat(string1,"Contact: ");
        strcat(string1,header[tcid].desde);
        strcat(string1,"@");
        strcat(string1,config.ipsalida);
        strcat(string1,":5060>");
        strcat(string1,crlf);
        strcat(string1,"Call-ID: ");
        strcat(string1,header[tcid].callid);
        strcat(string1,"@");
        strcat(string1,config.ip_local);
        strcat(string1,crlf);
        strcat(string1,"Max-Forwards: 70");
        strcat(string1,crlf);
        strcat(string1,"CSeq: ");
        strcat(string1,"2");
        strcat(string1," BYE");
        strcat(string1,crlf);
        strcat(string1,"From: ");
        strcat(string1,header[tcid].desde);
        strcat(string1,"@");
        strcat(string1,config.proxy);
        strcat(string1,">;tag=");
        strcat(string1,header[tcid].tagdesde);
        strcat(string1,crlf);
        strcat(string1,"To: ");
        strcat(string1,header[tcid].para);
        strcat(string1,"@");
        strcat(string1,config.proxy);
        strcat(string1,">;tag=");
        strcat(string1,header[tcid].tagpara);
        strcat(string1,crlf);
        strcat(string1,"User-Agent: SJLabs-SJphone/1.30.229b");
        strcat(string1,crlf);
        strcat(string1,"Via: ");
        strcat(string1,header[tcid].via);
        strcat(string1,crlf);
        strcat(string1,crlf);
        Xspy(SPY_KBTEL, SPY_EVENT,"Long string1 de BYE es
%d",strlen(string1));
        if(status=SOCK_SendTo(SipSocket,string1,strlen(string1),5060,config.proxy
)!=SOCK_SUCCESS)
            Xspy(SPY_KBTEL, SPY_EVENT,"Error al mandar el BYE");
        else
            Xspy(SPY_KBTEL, SPY_EVENT,"BYE Enviado");
    break;
    case 1:
        strcpy(string1,"SIP/2.0 200 OK");
        strcat(string1,crlf);
        strcat(string1,"Via: ");
        strcat(string1,header[tcid].via);
        strcat(string1,crlf);
        strcat(string1,"From: ");
        strcat(string1,header[tcid].desde);
        strcat(string1,";tag=");
        strcat(string1,header[tcid].tagdesde);
        strcat(string1,crlf);
        strcat(string1,"To: ");
        strcat(string1,header[tcid].para);
```

```
                        strcat(string1,";tag=");
                        strcat(string1,header[tcid].tagpara);
                        strcat(string1,crlf);
                        strcat(string1,"Call-ID: ");
                        strcat(string1,header[tcid].callid);
                        strcat(string1,crlf);
                        strcat(string1,"CSeq: ");
                        strcat(string1,"1");
                        strcat(string1," BYE");
                        strcat(string1,crlf);
                        strcat(string1,"Contact: ");
                        strcat(string1,"<sip:");
                        strcat(string1,config.ip_local);
                        strcat(string1,":5060>");
                        strcat(string1,crlf);
                        strcat(string1,"Content-Type: application/sdp");
                        strcat(string1,crlf);
                        strcat(string1,"Content-Length:0");
                        strcat(string1,crlf);
                        if(status=SOCK_SendTo(SipSocket,string1,strlen(string1),5060,config.proxy
)!=SOCK_SUCCESS)
                            Xspy(SPY_KBTEL, SPY_EVENT,"Error al mandar el OK");
                        else
                            Xspy(SPY_KBTEL, SPY_EVENT,"200 OK Enviado");
                    break;
                }
            case 6:
                strcpy(string1,"CANCEL sip:");
                strcat(string1,config.ip_server);
                strcat(string1," SIP/2.0");
                strcat(string1,crlf);
                strcat(string1,"Content-Length: 0");
                strcat(string1,crlf);
                strcat(string1,"Call-ID: ");
                strcat(string1,header[tcid].callid);
                strcat(string1,crlf);
                strcat(string1,"Max-Forwards: 70");
                strcat(string1,crlf);
                strcat(string1,"From: ");
                strcat(string1,header[tcid].desde);
                strcat(string1,";tag=");
                strcat(string1,header[tcid].tagdesde);
                strcat(string1,crlf);
                strcat(string1,"CSeq: ");
                strcat(string1,"1");
                strcat(string1," CANCEL");
                strcat(string1,crlf);
                strcat(string1,"To: ");
                strcat(string1,header[tcid].para);
                strcat(string1,";tag=");
                strcat(string1,header[tcid].tagpara);
                strcat(string1,crlf);
                strcat(string1,"Via: ");
                strcat(string1,header[tcid].via);
                strcat(string1,";5060");
                strcat(string1,"branch=z9hG4bKc0a801bd0131c9b14276a47d000021a900000006");
                strcat(string1,crlf);
                strcat(string1," User-Agent: SJLabs-SJphone/1.30.229b");
                strcat(string1,crlf);
                strcat(string1,crlf);
                if(status=SOCK_SendTo(SipSocket,string1,strlen(string1),5060,config.proxy)!=SOCK_
SUCCESS)
                    Xspy(SPY_KBTEL, SPY_EVENT,"Error al mandar CANCEL");
                else
                    Xspy(SPY_KBTEL, SPY_EVENT,"CANCEL Enviado");
            break;
        }
}
```

4.5 SISTEMA FUNCIONANDO

4.5.1 MENSAJES GENERADOS Y RESPUESTAS



- Un usuario en el Distrito Federal levanta el auricular, se activa un POT con un puerto de voz y se escucha el dial tone.
- El dial tone durará 12 seg. antes de escuchar el busy tone.
- El busy tone durará 12 seg.
- Marca a un número celular del Distrito Federal 015515077541.
- Al presionar cada dígito se escuchará un tono y se validará el código antifraude.
- El tiempo de espera para marcación es de 12 seg.
- Al terminar los 12 seg. se envía INVITE, incluyendo su puerto de voz asignado.

INVITE

```
INVITE sip:015515077541@207.17.190.58 SIP/2.0
Content-Length: 116
Contact: <sip:6642315019@201.154.109.182:5060>
Call-ID: 8AFD42A0-2D92-40EA-A67A-26AE5D94DC41@192.168.1.190
Content-Type: application/sdp
From:
"Administrator"<sip:6642315019@207.17.190.58>;tag=623409300
CSeq: 1 INVITE
Max-Forwards: 70
To: <sip:015515077541@207.17.190.58>
Via: SIP/2.0/UDP 192.168.1.190:5060
User-Agent: SJLabs-SJphone/1.30.229b

v=0
o=- 3319322138 3319322138 IN IP4 201.154.109.210
s=-
c=IN IP4 201.154.109.210
t=0 0
m=audio 5000 RTP/AVP 8 0
```

- El servidor SIP captura el #telefónico destino (0155 15077541), contesta TRYING y 200 OK a la tarjeta que origina INVITE:

TRYING

```
SIP/2.0 100 Trying
To: <sip:015515077541@207.17.190.58>;tag=646637f08d
From: "Administrator"
<sip:6642315019@207.17.190.58>;tag=262515
Call-ID: D5683515-B7A1-472C-B83C-33371EAFD9F7@192.168.1.190
CSeq: 1 INVITE
Contact: "015515077541" <sip:015515077541@207.17.190.58:5060>
Via: SIP/2.0/UDP 192.168.1.190:5060
User-agent: Excel_CSP/82.30.183
Content-Length: 0
```

200 OK

```
SIP/2.0 200 OK
To: <sip:015515077541@207.17.190.58>;tag=646637f08d
From: "Administrator"
<sip:6642315019@207.17.190.58>;tag=262515
Call-ID: D5683515-B7A1-472C-B83C-33371EAFD9F7@192.168.1.190
CSeq: 1 INVITE
Contact: "015515077541" <sip:015515077541@207.17.190.58:5060>
Supported: timer
Session-Expires: 1800; refresher=uas
Via: SIP/2.0/UDP 192.168.1.190:5060
User-agent: Excel_CSP/82.30.183
Content-Type: application/sdp
Content-Length: 146
Record-Route: <sip:207.17.190.58:5060;lr>

v=0
o=sip 0 0 IN IP4 207.17.190.58
s=SIP_Call
c=IN IP4 207.17.190.58
t=0 0
m=audio 30626 RTP/AVP 0 101
a=rtpmap:101 telephone-event/8000
```

- La tarjeta de voz captura el puerto de voz del servidor SIP (30626) y le contesta ACK:

ACK

```
ACK sip:015515077541@207.17.190.58:5060 SIP/2.0
Content-Length: 0
Contact: <sip:6642315019@201.154.109.182:5060>
Call-ID: 8AFD42A0-2D92-40EA-A67A-26AE5D94DC41@192.168.1.190
Max-Forwards: 70
CSeq: 1 ACK
From: <sip:6642315019@207.17.190.58>;tag=623409300
To: <sip:015515077541@207.17.190.58>;tag=8942968
User-Agent: SJLabs-SJphone/1.30.229b
Via: SIP/2.0/UDP 192.168.1.190:5060
```

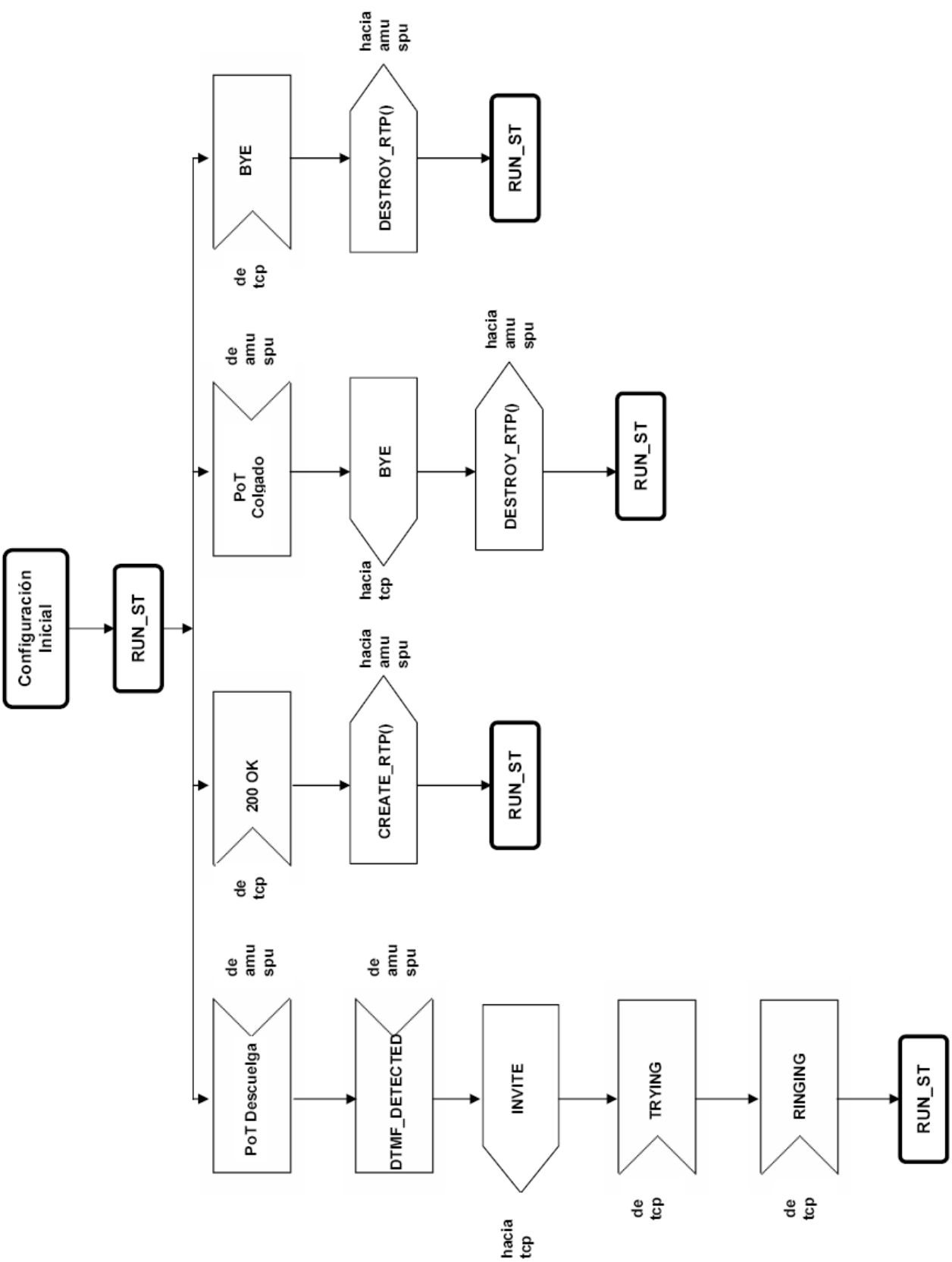
- Se generan los paquetes de voz RTP entre el servidor SIP y la tarjeta de voz, utilizando los puertos de voz específicos (5000 y 30626), el servdor enlaza la llamada hacia una central telefónica de la red fija o móvil.
- Cuando alguna de las partes cuelga el teléfono, se termina la conexión de voz. La parte que cuelga envía un BYE a la otra, en nuestro caso la tarjeta del gateway cuelga.

BYE

```
BYE sip:015515077541@207.17.190.58:5060 SIP/2.0
Content-Length: 0
Contact: <sip:6642315019@201.154.109.182:5060>
Call-ID: 8AFD42A0-2D92-40EA-A67A-26AE5D94DC41@192.168.1.190
Max-Forwards: 70
CSeq: 2 BYE
From: <sip:6642315019@207.17.190.58>;tag=623409300
To: <sip:015515077541@207.17.190.58>;tag=8942968
User-Agent: SJLabs-SJphone/1.30.229b
Via: SIP/2.0/UDP 192.168.1.190:5060
```

200 OK

```
SIP/2.0 200 OK
To: <sip:015515077541@207.17.190.58>;tag=646637f08d
From: <sip:6642315019@207.17.190.58>;tag=262515
Call-ID: D5683515-B7A1-472C-B83C-33371EAFD9F7@192.168.1.190
CSeq: 2 BYE
Via: SIP/2.0/UDP 192.168.1.190:5060
User-agent: Excel_CSP/82.30.183
Content-Length: 0
Record-Route: <sip:207.17.190.58:5060;lr>
```


Configuración Inicial
RUN_ST
PoT Descuelga
de amu spu
DTMF_DETECTED
de amu spu
INVITE
hacia tcp
TRYING
de tcp
RINGING
de tcp
RUN_ST
200 OK
de tcp
CREATE_RTP()
hacia amu spu
RUN_ST
PoT Colgado
de amu spu
BYE
hacia tcp
DESTROY_RTP()
hacia amu spu
RUN_ST
BYE
de tcp
DESTROY_RTP()
hacia amu spu
RUN_ST

CAPÍTULO 5
MERCADO

5.1 TELEFONÍA EN MÉXICO

5.1.1 MARCO HISTÓRICO

Desde el establecimiento de la telefonía en nuestro país se han tenido fechas que han marcado el rumbo de este servicio hasta lo que es ahora. Por tal razón, se listan los acontecimientos más importantes a lo largo de su desarrollo:

- Diciembre de 1878, se establece oficialmente el servicio telefónico en México por Alfred Westrup & Co. para comunicar a las comisarías de policía, la inspección general, la oficina del gobernador de la ciudad y el ministerio de gobernación.

- Julio de 1882, se forma la primera compañía telefónica mexicana conocida como Mextelco.

- Durante los años 1883 y 1898 se instaló un conmutador múltiple completo, con una capacidad de dos mil líneas.

- Octubre de 1904, la empresa de Teléfonos Ericsson, S.A., inauguró su servicio con 300 suscriptores, construyendo para 1911, las líneas a Tlalnepantla y Cuautitlán.

- Febrero de 1905, la Compañía Telefónica Mexicana aumentó su capital y cambió de nombre a Compañía Telefónica y Telegráfica Mexicana, S.A.

- En 1924, Ericsson inauguró la primera central telefónica automática conocida como la Central Roma con una capacidad de diez mil líneas.

- En 1926 se retira la intervención gubernamental que desde 1915 padecía la Compañía Telefónica y Telegráfica Mexicana, S.A., adquiriéndola la empresa International Telephone and Telegraph Co. (ITT).

- Septiembre de 1927, se inicia con el servicio de larga distancia (LD), realizándose la primer conferencia telefónica entre el general Plutarco Elías Calles y Calvin Coolidge (EE.UU.). Dos meses después se tuvo comunicación con Canadá y un año más tarde con Europa.

- Agosto de 1947, Ericcson deja de operar en México después de 50 años

- Diciembre de 1947, se constituyó Teléfonos de México, S. A. (Telmex).

- Entre 1961 y 1962, según la publicación *The Word's Telephone*, México ocupa el séptimo lugar a nivel mundial en cuanto a desarrollo tecnológico y el primero en el continente americano.

- Verano de 1962, es lanzado el satélite de comunicaciones *Telstar*, patrocinado por el sistema Bell y la NASA; siendo el primero en funcionar con el sistema de microondas, lo cual permitió que las ciudades de México, Monterrey y Nuevo Laredo mejoraran su servicio de conmutación automática de larga distancia.

- En 1970, el sistema de telecomunicaciones mexicano queda comunicado con más de mil líneas en el Distrito Federal, 334 en Guadalajara, 291 en León, 247 en Toluca y 247 en Puebla, para la transmisión del Campeonato Mundial de Fútbol,

celebrado en México; evento para el que, además, se contó con 100 casetas de larga distancia instaladas en los centros de prensa y 129 líneas privadas para el uso de télex y teléfono. Al mismo tiempo, se añadió en el Valle de México un dígito a los números telefónicos, se antepuso el número 5 a los existentes, para llegar a siete cifras.

- Durante el año de 1970, Se conectaron 39 circuitos los cuales permitirían a México comunicarse directamente con Argentina, Brasil, Colombia, Chile, España, Francia, Inglaterra, Italia, Japón, Panamá, Perú y Venezuela.

- Agosto de 1972, el gobierno federal adquiere el 51 por ciento de las acciones del capital social de Teléfonos de México, por lo que dejó de ser privada y pasó a tener participación estatal mayoritaria.

- Junio de 1980, Teléfonos de México se incorpora al uso de sistemas digitales, lo cual representa ventajas considerables ya que estos sistemas presentan menor sensibilidad a distorsión e interferencia, mayor facilidad en la conmutación, menor espacio para el equipo digital, además de la posibilidad de transmitir varios canales por un mismo circuito (30 por cada dos pares telefónicos).

- Durante el año e 1982, Telmex pone en operación las instalaciones de enlaces por fibras ópticas.

- Agosto de 1985, entra en operación el satélite Morelos I.

- Durante el año de 1987, se puso en servicio la central de Red Digital de Servicios Integrados (RDSI), la cual permite que los usuarios utilicen en forma simultánea una línea telefónica digital y transmitir los servicios de voz, datos, vídeo y fácilmente.

- En 1990, se inicia un proceso de privatización de las empresas que pertenecían al estado (bancos, teléfonos, agricultura, etc.) en búsqueda de mayor desarrollo e inversión, con lo cual se puso en venta Telmex con una ventaja de seguir siendo monopolio por 7 años más. Después de su privatización Telmex comenzó con un plan de inversión en nueva tecnología, fibra óptica, y cobertura total del país.

- A principios de los 1990 comenzaron a surgir muchas compañías que ofrecían servicios de telefonía móvil, para 1993 Iusacell se había convertido en el líder tras comprar varios operadores regionales. Telmex no tenía inversiones en este negocio, así que decidieron entrar al mercado con la empresa Radio Móvil Dipsa, con la marca Telcel que estaba en un lejano segundo lugar en el mercado nacional, cuando Iusacell contaba con 3,000,000 de usuarios, Telcel tenía menos de 1,000,000. Este escenario cambio en 1995 cuando México sufrió una de sus peores crisis económicas, Iusacell decidió enfocarse en clientes de alto nivel con planes de renta mensual de alto costo, y Telcel decidió enfocarse al sector de menores ingresos con los primeros planes de prepago. Esto causó que Telcel obtuviera más clientes y se convirtiera en el líder en el mercado nacional teniendo el doble de usuarios que Iusacell dos años después.

- A partir de 1992, Teléfonos de México inicia un proyecto para facilitar la prestación de servicios de comunicación de voz, datos y de imágenes, que actualmente se prestan en redes independientes. Al proyecto se le denominó Red Digital Superpuesta, su infraestructura es de alta tecnología, por lo que se creó una red especial que está superpuesta a la red telefónica existente; se inició su operación

en México, Monterrey y Guadalajara, con 25 mil troncales digitales de alta velocidad para conmutadores telefónicos. Se tendieron 400 kilómetros de cables de fibra óptica y 17 sistemas de radiocomunicación digital por microondas, lo cual permitió enlazar a 40 centros de acceso a la red distribuidos en estas tres ciudades.

- En 1997 se abrió el mercado mexicano de la telefonía, con lo cual entraron AT&T, MCI y Axtel, entre otras, pero ninguna logró afectar seriamente a Telmex.

- Durante el periodo comprendido entre el año de 1990 a diciembre del 2004, Telmex ha experimentado uno de los mayores crecimientos en la industria de las telecomunicaciones a nivel mundial. En la Tabla I se presentan algunas cifras que muestran dicho crecimiento.

- En resumen, Telmex cuenta con tres veces más líneas telefónicas que en 1990. En telefonía pública el incremento fue de 7.9 veces. Actualmente existen 2,576 poblaciones con acceso a Internet. En dicho periodo la telefonía creció tres veces por encima de la economía nacional. El acceso al servicio telefónico superó los 86 millones de habitantes en 2003, esto es un 45% adicional con respecto a 1990

CONCEPTO	1990	2004
Líneas Telefónicas	5,352,824	17,172,278 (*)
Teléfonos Públicos	92,073	725,483 (**)
Cuentas de Internet		1,741,296 (*)
Poblaciones con servicio	10,621|	20,848 (*)
Tráfico Local (millones de llamadas)	8,950	26,782 (*)
Tráfico de Larga Distancia Nacional (Millones de min.)	4,375	16,700 (*)
Digitalización de la Red	30.9%	100% (**)
Número de concesionarios interconectados	10	20 (*)

NOTA
(*) Cifras a septiembre de 2004.
(**) Cifras a diciembre de 2003.

Tabla I. Crecimiento de TELMEX en el periodo 1990-2004

5.1.2 PANORAMA ACTUAL

Las redes de telecomunicaciones respecto al punto de vista del servicio que ofrecen pueden ser clasificadas en tres tipos:

1) Red de Telefonía Pública Conmutada

Uno de los principales problemas que tiene la red local de acceso tradicional en cuanto a su rentabilidad, es el hecho de que necesita tener asignado en forma permanente un puerto en la central telefónica, un par físico de alto costo, los componentes auxiliares y los elementos compartidos como: protectores de línea, distribuidor, gabinetes, fuentes de poder, “software”, posteos, pozos, ductos, etc.

Estas características intrínsecas de la red local, hacen que mientras en ésta el costo es directamente proporcional a la cantidad de abonados, en la larga distancia, el costo por circuito y kilómetro es inversamente proporcional a la cantidad de circuitos en uso, de un determinado enlace.

Aunque parezca paradójico, al formarse las nuevas telefónicas locales, el desarrollo del negocio se daba fundamentalmente en los servicios de acceso; es decir, lo que se cobraba a los operadores de LD por el mismo. En tanto que la telefonía local no representaba un negocio rentable por sí mismo, las compañías locales tuvieron que desarrollar otros negocios; se les autorizó la venta y manufactura de equipo telefónico terminal, y se desarrollaron servicios de valor agregado, como la llamada en espera, el desvío, la conferencia, etc., con la finalidad de que mediante la aplicación de facilidades basadas en “software”, se obtuviera una mejor ganancia de la infraestructura.

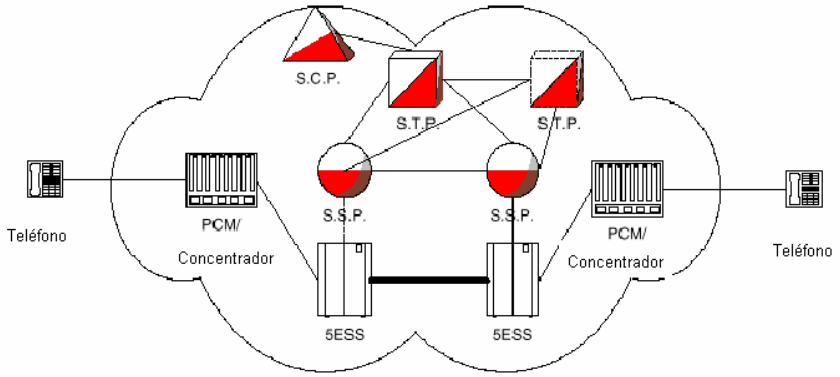


Figura 5.1 Arquitectura convencional de una red PSTN

2) Red Inalámbrica

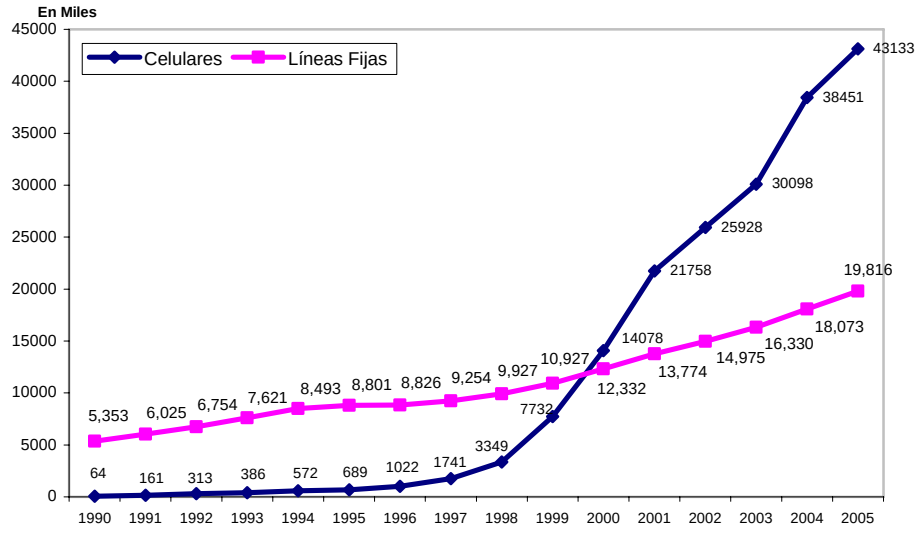
Este tipo de redes actualmente se encuentra en un proceso de convergencia con redes de datos, que permite integrar servicios diversos dentro de un mismo dispositivo final.

Así mismo se vislumbra que la tecnología empleada en la red celular sea complementada o integrada con la tecnología inicialmente diseñada para el acceso a datos en forma inalámbrica como lo son WiFi o Blue tooth (Fig. 5.2).

Figura 5.2. Servicios inalámbricos para voz y datos

En los últimos quince años, la red de telefonía móvil celular ha tenido un avance espectacular marcado por cuatro generaciones tecnológicas (primera generación analógica con modulaciones en FM, segunda generación digital basada en TDMA y CDMA, sistema GSM (conocido también como de segunda generación y media) y tercera generación que ofrecerán servicios multimedia y multitransmisión a altas velocidades (2 Mbps) de voz y datos.

A esta intensa evolución tecnológica ha correspondido un crecimiento también intenso. En quince años, el número de líneas móviles supera por más del doble al número de líneas fijas en todo el mundo. Tan sólo en México, al cierre del 2004, mientras que el número de líneas fijas era de alrededor de 17 millones, el número de celulares fue de 38 millones. En la Fig. 5.3 se muestra la gráfica comparativa del crecimiento de líneas celulares vs. Líneas fijas.



Fuente: COFETEL

Figura 5.3. Gráfica comparativa de crecimiento de líneas celulares vs. líneas fijas.

3) INTERNET.

Es la red mundial conformada por diferentes redes regionales que transmiten datos, mediante una secuencia de “paquetes” los cuales contienen información.

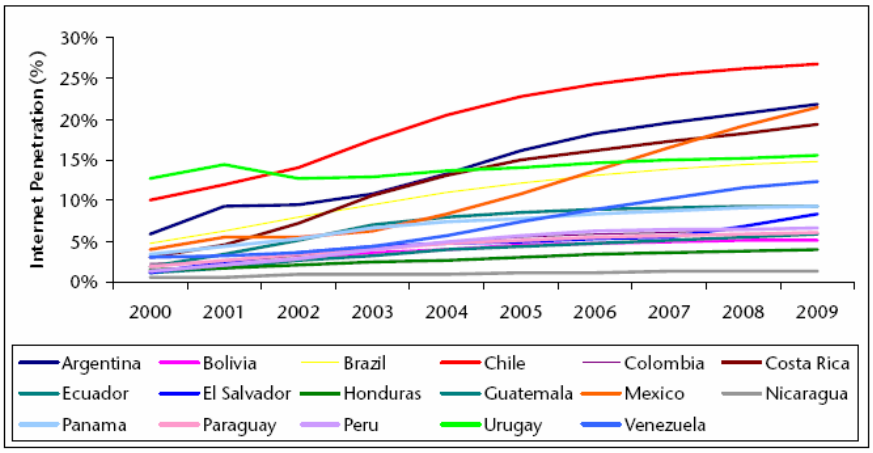
La red de Internet está diseñada para transmitir datos sobre toda una gama de medios de comunicación. Es por lo tanto, sumamente flexible. A Internet se puede acceder mediante la PSTN utilizando un módem o mediante un acceso dedicado

de banda ancha, ya sea por radio, fibra óptica, ADSL, HFC o cable coaxial. El prestador de servicios Internet (ISP) es interconectado con otros mediante dispositivos de enrutamiento o enrutadores, los cuales permiten conmutar cada paquete según la dirección de destino.

Utilizando las aplicaciones de la terminal del usuario (por ejemplo, la computadora) cualquier información puede convertirse en datos y enviarse como paquete sobre Internet, ya sea voz, imágenes, video o datos.

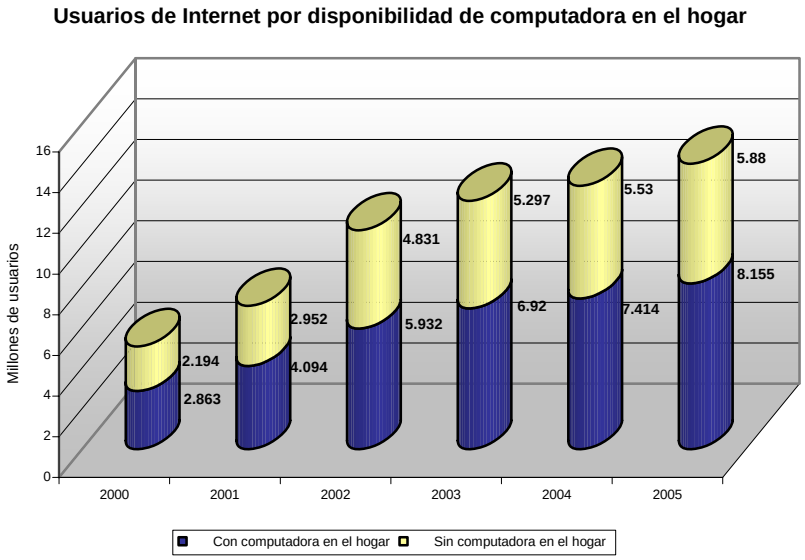
Al convertirse en bits, la voz puede ser transmitida por Internet como paquetes de datos. Este es el principio básico de VoIP.

Actualmente Internet incluye aproximadamente 5000 redes en todo el mundo y más de 100 protocolos distintos basados en TCP/IP, que se configura como el protocolo de la red. Los servicios disponibles en la red mundial han avanzado mucho gracias a las nuevas tecnologías de transmisión de alta velocidad, como DSL y Wireless. El 3 de enero de 2006 Internet alcanzó los mil millones de usuarios. En la Fig. 5.4 se observa la gráfica de un estudio de Pyramid Research de diciembre de 2004, mostrando la creciente tasa de penetración de Internet en varios países de América Latina. Esta curva presenta el enorme potencial de esta red. Se prevé que en diez años, la cantidad de navegantes de la Red aumentará a 2,000 millones. La Fig. 5.5 muestra los usuarios de Internet por disponibilidad de computadora en el hogar, como puede observarse, la penetración de este medio es muy grande en aquellos usuarios que no disponen de una computadora en el hogar.



Fuente: Pyramid Research.

Figura 5.4. Tasa de penetración de Internet en América Latina

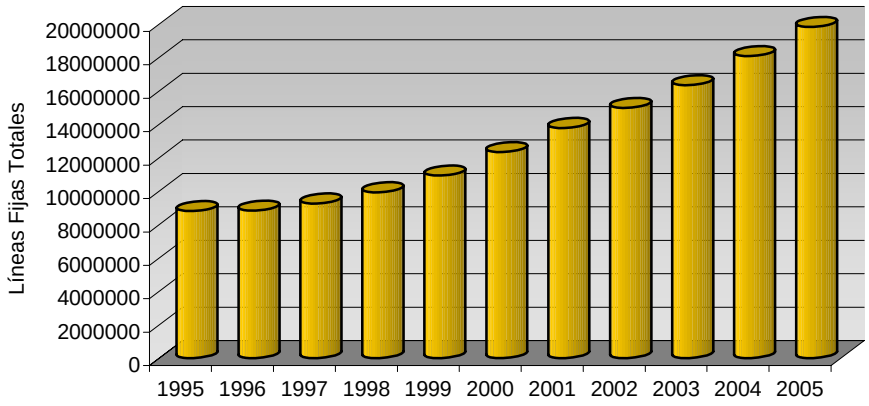


Fuente: Dirección General de Tarifas e Integración Estadística, COFETEL, con información del INEGI.
Figura 5.5 Usuarios de Internet por disponibilidad de computadoras en casa.

5.1.3 DESCRIPCIÓN DEL MERCADO DE TELEFONÍA FIJA

5.1.3.1. TELEFONÍA LOCAL

El proceso de privatización de Telmex en 1990, fue el punto de partida para transitar de un esquema monopólico hacia uno de competencia. En la Fig. 5.6 se muestra el crecimiento de la telefonía fija en el periodo 1995-2005. Los cambios que en ese momento se introdujeron a título de concesión permitieron establecer compromisos específicos sobre expansión de la red y calidad del servicio. En telefonía básica se pasó de 5.4 millones de líneas en 1990 a 19.8 millones de líneas en 2005, con lo que a escala nacional, la teledensidad se incrementó en ese mismo periodo de 6.4 líneas por cada 100 habitantes a 19.8 por cada 100 habitantes. La Fig. 5.7 muestra la gráfica de la teledensidad en comparación con el ingreso per cápita para varias ciudades del país.



Nota: Cifras revisadas desde 2000. A partir de 1999, Incluye a los nuevos concesionarios de telefonía local.
FUENTE: Dirección General de Tarifas e Integración Estadística, COFETEL.
Fig 5.6. Crecimiento de la Telefonía Fija

En el área de telefonía local además de las concesiones otorgadas para proporcionar el servicio por medios alámbricos e inalámbricos, se han consolidado las áreas de servicio local de 1,464 a 406. Así mismo se dio el crecimiento de la numeración nacional y se insertaron las nuevas claves de larga distancia para todo el país.

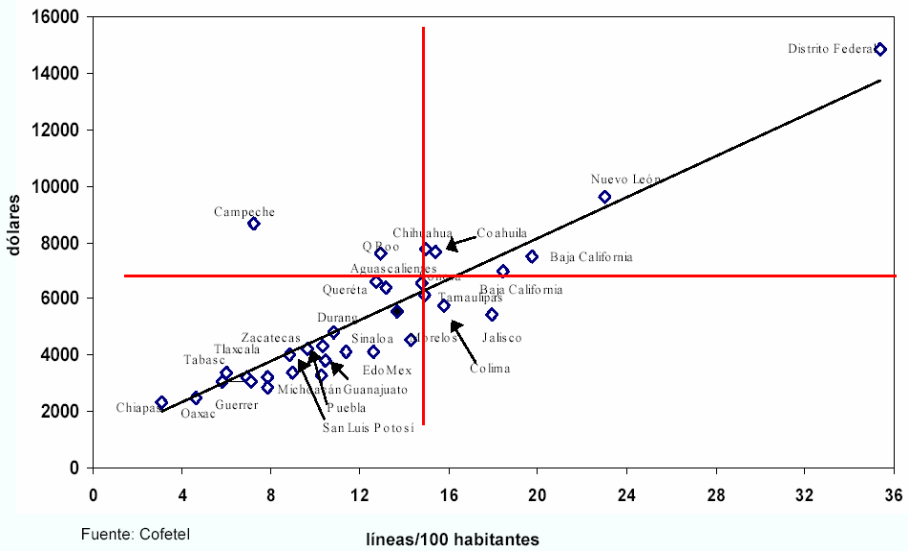


Figura 5.7. Teledensidad Fija Vs. Ingreso Per Cápita (US$)

5.1.3.2. TELEFONÍA LARGA DISTANCIA

A partir de 1995, se otorgaron los primeros títulos de concesión para prestar el servicio telefónico de larga distancia, de forma tal que las empresas entrantes contaran con un tiempo adecuado para desplegar sus redes e iniciar la prestación de sus servicios a partir del 1 de enero de 1997.

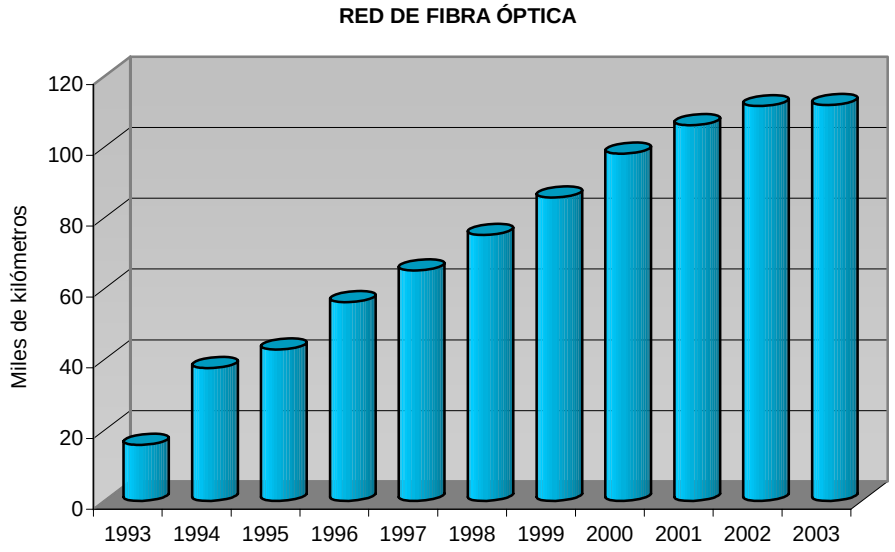
Algunos indicadores de los primeros tres años de la apertura fueron:

- Las tarifas de larga distancia se redujeron en más del 50% en términos reales,
- El tráfico ha crecido a una tasa anual promedio de 21%; y,
- Hasta el mes de mayo de 2000, se han presentado 13 millones 689 mil solicitudes de cambios válidos de operador por parte de los usuarios.

La competencia en el servicio de larga distancia ha impulsado también la introducción de nuevos servicios como redes privadas virtuales, transmisión de datos con nuevos protocolos y servicios complementarios.

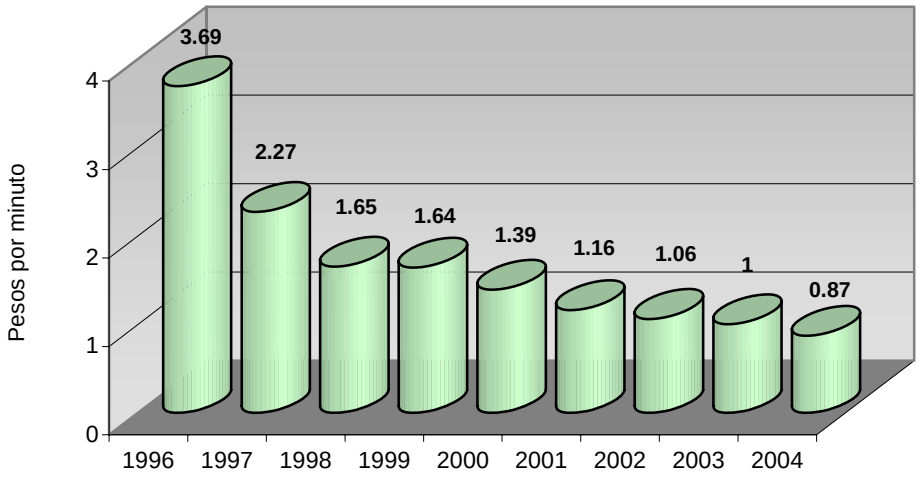
5.1.3.3. LARGA DISTANCIA NACIONAL

En los últimos cinco años, la red de fibra óptica se duplicó, pasando de 42,765 a 85,196 kilómetros, siendo las nuevas empresas concesionarias las que contribuyeron en 50 por ciento a este incremento (23,103 kilómetros), lo que ha permitido que el usuario cuente ahora con una mayor calidad y diversidad de servicios. En la Fig. 5.8 se muestra la gráfica de dicho incremento.



Nota: A partir de 1996 incluye la red de los nuevos operadores de larga distancia y a partir de 1999 la de los nuevos concesionarios de telefonía local.
FUENTE: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.
Figura 5.8.Incremento de la red de fibra óptica.

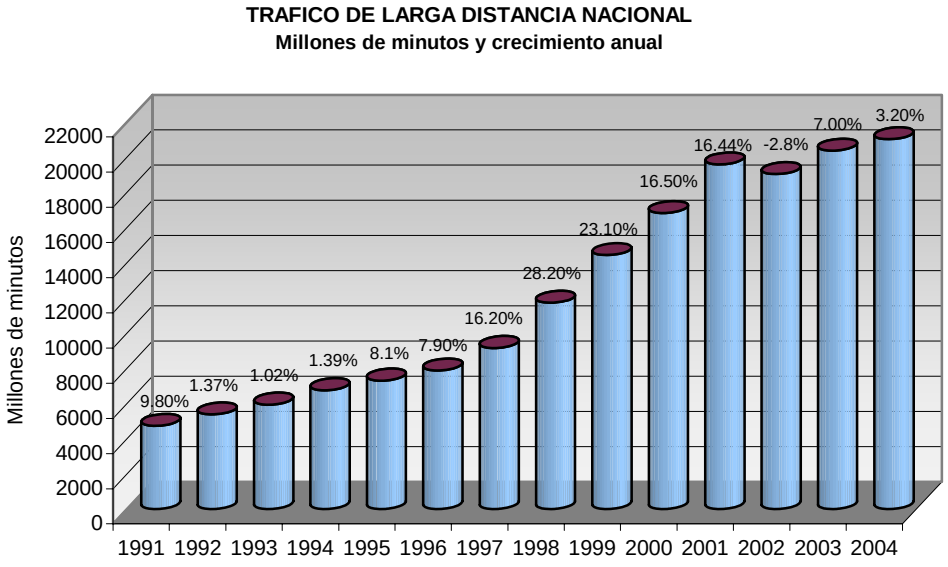
En términos reales, en México desde el inicio de la competencia en 1997, las tarifas al público del servicio telefónico de larga distancia nacional se han reducido en 60 por ciento. Tal como puede verse en la Fig. 5.9.



FUENTE: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.
Figura 5.9. Gráfica de la reducción de tarifas de larga distancia.

Esto se ha traducido en incrementos constantes del tráfico de este servicio. En el periodo 1997–1999, el tráfico de larga distancia nacional, medido en minutos, se elevó a una tasa promedio anual de 22.4 por ciento. Este incremento se muestra en la Fig. 5.10.

Otro factor es que Telmex, con la reducción de las áreas de servicio local, eliminó el 71% de los destinos de larga distancia, beneficiando con esta medida a unos 25 millones de usuarios.



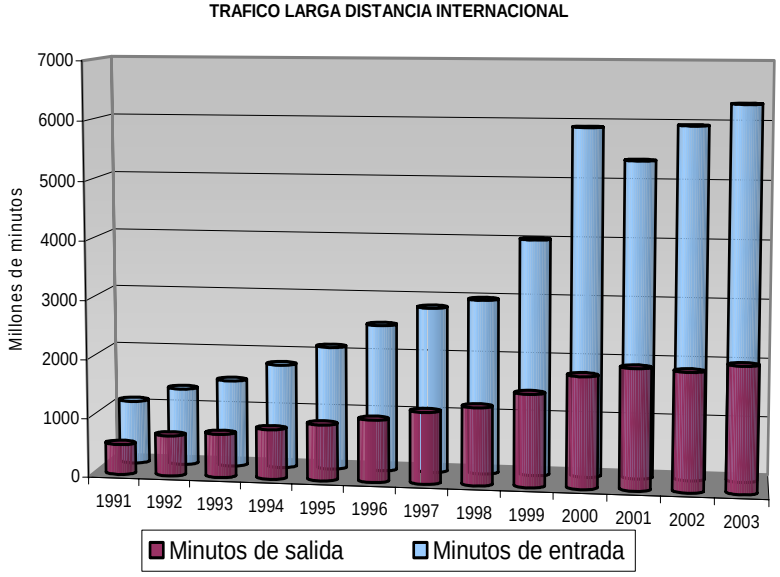
Fuente: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.

Figura 5.10 Gráfica del tráfico de LDN.

5.1.3.4. LARGA DISTANCIA INTERNACIONAL

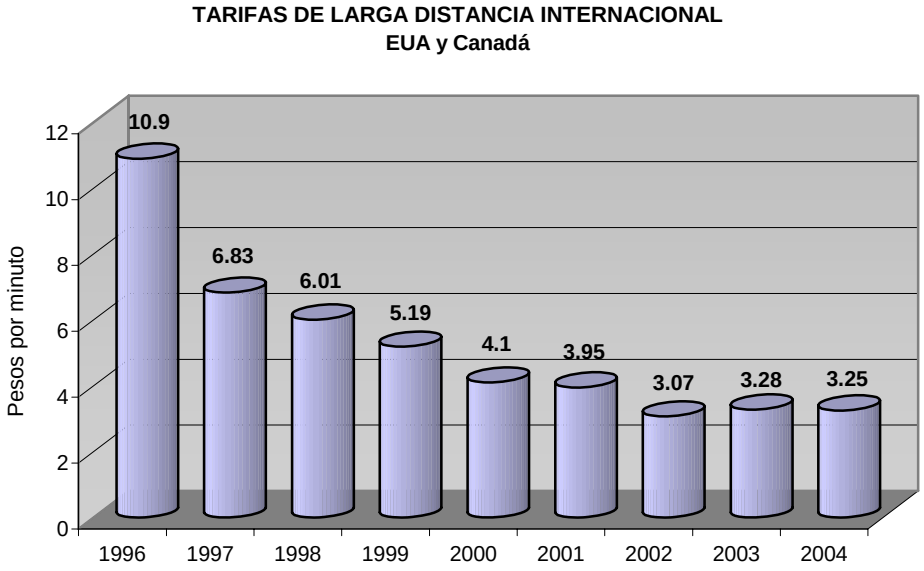
El 1 de enero de 1997, al igual que el servicio telefónico de larga distancia nacional se abrió a la competencia el servicio telefónico de larga distancia internacional conforme se tenía previsto desde la privatización de Telmex.

Para permitir la apertura de la larga distancia internacional a la competencia, en los meses de enero y febrero de 1997, el pleno de la Comisión Federal de Telecomunicaciones COFETEL inició las autorizaciones provisionales de convenios de interconexión con operadores extranjeros y puertos internacionales a cuatro concesionarios de larga distancia, de conformidad con los requisitos establecidos en las RLDI (Reglas para prestar el Servicio de Larga distancia internacional). En la Fig. 5.11 se muestra el tráfico de la larga distancia internacional.

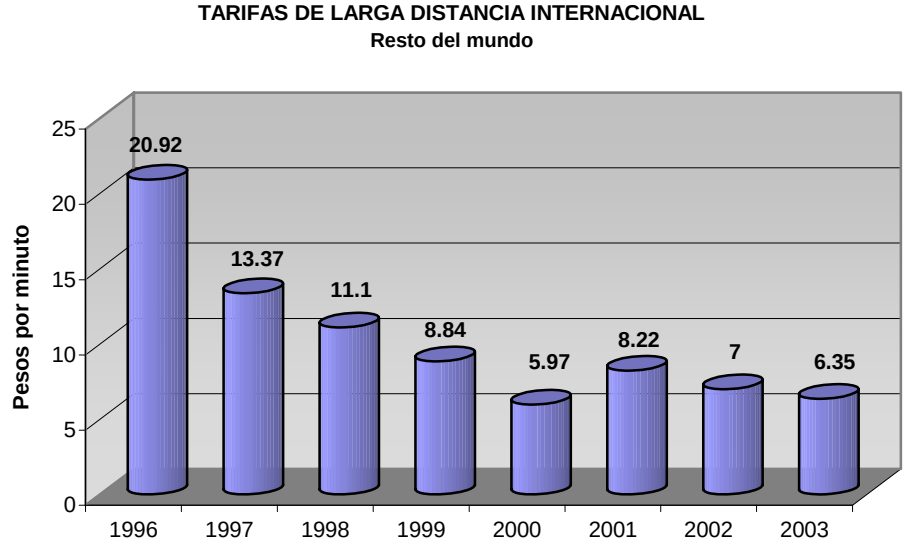


Fuente: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.
Figura 5.11 Gráfica del tráfico de larga distancia internacional

De igual forma que con la Larga Distancia Nacional, las tarifas en este servicio tuvieron un importante decremento a partir de la apertura del mercado. En la Fig. 5.12 y en la 5.13 se muestran las gráficas de la reducción de tarifas a Estados Unidos y Canadá, y a el resto del mundo, respectivamente.



Fuente: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.
Figura 5.12 Tarifas de LDI en EUA y Canadá



Fuente: Dirección General de Tarifas e Integración Estadística, **COFETEL**, con información proporcionada por los concesionarios.
Figura 5.13 Gráfica de tarifas de LDI en el resto del mundo

5.1.3.5. TELEFONÍA SOBRE IP BASADA EN VOIP

Las redes de Voz sobre IP (VoIP) intentan sustituir a los operadores telefónicos tradicionales al tratar de reducir costos, esta reducción es posible ya que cuentan con una misma plataforma para voz y datos, de esta forma pueden ofrecer una amplia gama de servicios relacionados a la telefonía, todo esto bajo el protocolo IP.

Esta tecnología resulta sumamente atractiva para las empresas, no solo por la considerable reducción de costos, sino por el manejo de los servicios. Así, si consideramos que en cualquier oficina moderna tanto la comunicación hacia adentro de la empresa como hacia afuera de ella representa una necesidad vital, independientemente del tipo de negocio que se trate. Estas oficinas contienen teléfonos, faxes, computadoras con conexión a Internet, laptops con conexiones inalámbricas y teléfonos celulares, entre otros equipos. Dada la proliferación de dispositivos, se busca la forma tanto de unificarlos así como de manejarlos eficientemente.

Hace no muchos años, era una novedad utilizar Internet para realizar una llamada telefónica, sin embargo, en la actualidad esto es una realidad cotidiana. De acuerdo con George Goodall, analista de InfoTech, la adopción de VoIP se ha dado más rápido de lo que se pensaba. La encuesta de enero de 2005 de InfoTech muestra que el 23% las compañías de América del norte y de Europa occidental cuyo capital es menor a los dos mil millones de dólares, han hecho alguna clase de investigación sobre VoIP. Un 16% planeó hacer inversiones en el 2005 y un 24% dijo que haría inversiones en VoIP durante los siguientes tres años. A finales de 2008 se proyecta que el 40% de las compañías adoptará alguna forma de VoIP.

Para los vendedores de servicios el negocio resulta bueno. Alex Hadden-Boyd, directora de mercadotecnia de Cisco, considera que en el futuro será difícil comprar otra cosa que no sea telefonía sobre IP. Mientras esta telefonía empezó en universidades, gobiernos locales y estatales y largas instituciones financieras. La telefonía sobre IP está ahora inmersa en industrias de todos los tamaños alrededor del mundo, desde compañías de 15 empleados hasta compañías de decenas de miles de empleados. Ella menciona que entre los clientes de Cisco, más de sesenta poseen 5,000 o más teléfonos IP. En el caso de Nortel cerca de dos tercios de sus empleados utilizan teléfonos IP.

Un indicador para la compañía Cisco de cómo la tecnología VoIP ha entrado al mercado es que entre el 35 y 50% de de los envíos de sistemas telefónicos a las empresas y pequeños y medianos negocios son basados en IP. Las ventas de Cisco de teléfonos IP muestran la tendencia ascendente y la velocidad con la que se está adoptando esta tecnología al registrar más de 5 millones de envíos de teléfonos IP. Puesto en un contexto, mientras el primer millón se envió en un periodo de tres años y medio, los cuatro millones restantes se enviaron en cinco meses. Además, Cisco está desplazando cerca de 10,000 teléfonos tradicionales cada día laborable. En opinión de Tony Rybezynsky, Director de estrategia de tecnologías empresariales de Nortel, las compañías están adoptando la tecnología por muchas razones, más que solamente hacer una llamada telefónica, ya que la telefonía IP da mucha flexibilidad al virtualizar el ambiente telefónico, es decir, mientras que un conmutador se ubica en un cuarto y a él están conectados los dispositivos, la telefonía IP permite al usuario estar ubicado en cualquier lugar y que los recursos estén ubicados remotamente.

Según un informe de telefonía sobre IP de la UIT en 2001 se dice: “Se ha llevado a cabo una modificación del paradigma fundamental en la industria de las telecomunicaciones, que ha traído como consecuencia un cambio en las comunicaciones personales como ocurrió con el teléfono comparado con el telegrama. Este cambio consiste en el tránsito de las redes tradicionales de voz por RTC pública a las redes de datos por conmutación de paquetes, usando la tecnología IP.”

5.1.4 DESCRIPCIÓN DEL MERCADO DE TELEFONÍA BASADO EN VOZ SOBRE IP

Las soluciones VoIP proporcionan el potencial para ahorrar de un 50 a un 80% en costos de larga distancia en aplicaciones punto a punto. Los negocios con oficinas remotas alrededor del mundo o con un código de área diferente a cientos de kilómetros de distancia, pueden beneficiarse de una solución basada en VoIP. Por ejemplo, un negocio con una oficina central en Monterrey y algunas oficinas remotas puede rápidamente pagar por la solución VoIP y obtener sustanciales ahorros en cargos de larga distancia.

Con la interfaz apropiada, una solución VoIP se conectará directamente al equipo de teléfono o fax (Fig. 5.14), permitiendo a los negocios de todos tamaños agregar fácilmente voz sobre IP. La primera de las tres interfaces es llamada FXS (estación de intercambio remota), la cual se conecta directamente a teléfonos o faxes. La interfaz FXO (oficina de intercambio remota) se conecta a un PBX y proporciona accesos externos mientras que la interfaz E&M (interfaz de un dispositivo VoIP) se conecta a las líneas troncales de un PBX.

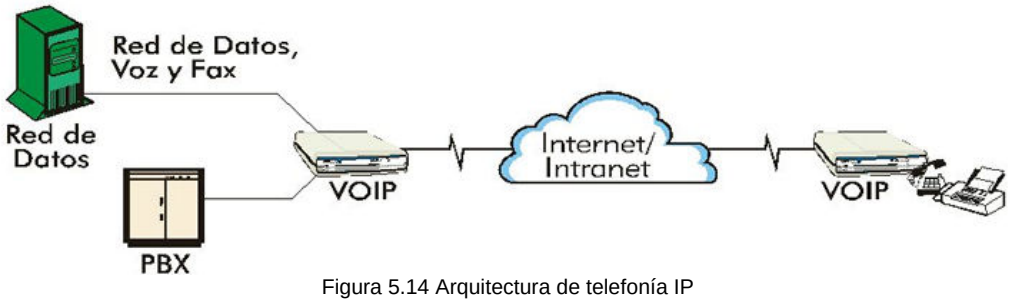


Figura 5.14 Arquitectura de telefonía IP

Entonces, las necesidades de comunicación en las empresas pueden satisfacerse mediante este esquema tal como se describe a continuación:

- Comunicación sucursal a sucursal

  Las oficinas de una sucursal de una compañía en la Ciudad de México y otra ubicada en el extranjero, comparten el mercado de información diariamente. Estas pequeñas oficinas conectan sus teléfonos y faxes directamente a un producto de VoIP.

- Comunicación corporativo a corporativo

  Diariamente las comunicaciones entre la oficina central de una compañía y un socio en el extranjero son críticas. Cada sitio conecta un producto VoIP a su PBX. Las llamadas son enrutadas sobre sus conexiones de Internet dedicadas. Hacer una llamada libre de recargos es tan fácil como marcar una extensión de la persona o máquina de fax en cualquier otro sitio.

- Comunicación sucursal a corporativo

  Una oficina de la sucursal hace con frecuencia llamadas importantes a un gerente de cuentas en la oficina central en otro país. Entonces, un teléfono y fax están conectados a un dispositivo VoIP. En dicha oficina central una extensión de PBX está conectada a un dispositivo VoIP.

Dado lo anterior, en el mercado de telefonía IP se estima un crecimiento importante como el que se describe a continuación:

- De acuerdo a un estudio internacional de la empresa IDC, entre el 2 y el 3% de los más de 400 millones de puertos utilizados en empresas a nivel mundial están habilitados para telefonía IP. No obstante, se espera que esta tecnología crezca a una tasa del 20% anual en los próximos cinco años.

- Un informe de Evalueserve advierte que los operadores fijos, especialmente los de larga distancia, serán los que más resentirán la competencia de VoIP, ya que los mayores esfuerzos se concentran hacia ese mercado.

- Entre los proveedores más activos en servicios de telefonía IP en México están:
  - Avantel
  - Alestra
  - Maxcom
  - Iusacom
  - Vonage
  - IP Matrix
  - MTM Telecom

- En términos de plataforma tecnológica, Avaya y Nortel son las mejor posicionadas en el mercado nacional, sin embargo Cisco Systems se encuentra penetrando fuertemente con soluciones integrales basadas en VoIP.

5.1.4.1. ANÁLISIS COMPETITIVO

Con base en las tendencias mostradas en la sección anterior, el mercado de telefonía IP muestra una importante competencia entre las compañías telefónicas y se vislumbra la participación de otros sectores como el de televisión por cable. Varios de los principales operadores del país han iniciado una guerra de soluciones convergentes (servicios de voz, datos y video) al introducir al mercado residencial y de pequeñas y medianas empresas PYMES, paquetes de telefonía IP – llamadas locales y de LD ilimitadas –, Internet de banda ancha y servicio de televisión por cable. A continuación se listan las acciones tomadas por estas compañías para satisfacer este mercado:

Telmex

Telmex, el operador dominante en telefonía fija, ha centrado su estrategia en la oferta de conectividad a Internet de banda ancha impulsada por planes de financiamiento (a dos y tres años) para la adquisición de computadoras. Por otro lado, ante la inminente disminución de los ingresos por concepto de larga distancia, la telefónica se ha visto en la necesidad de desarrollar nuevas soluciones y servicios.

IUSACom VoxIP

En noviembre de 2004 inició operaciones en México la empresa VOX IP. VoxIP es parte del grupo Iusa (Fig. 5.15). En 2003, Iusa compró VPN, una telefónica que poseía solo una concesión. Entre la compra de VPN y dotarse de una infraestructura básica, Iusa invirtió sólo 15 millones de dólares, a cambio de eso en 2005 calcula alcanzar 60,000 usuarios. Aunque VoxIP no tiene una red propia, sus márgenes de ganancia pueden ser del 60% por año.

Fig. 5.15 Servicio VoxIP de IusaCom

Avantel y MVS, NetVoice

En febrero de 2005, Avantel junto con la compañía MVS ofrecen el servicio Netvoice. Avantel tiene una red propia totalmente digitalizada de más de 8,000 kilómetros. Su oferta de VoIP ofrece acceso inalámbrico a la red de banda ancha. Avantel espera cerrar el 2005 con 60,000 clientes en VoIP (Fig. 5.16).

Fig.5.16 Servicio NetVoice de Avantel-MVS

Alestra, Masternet

Alestra, en un intento por recuperar participación en el segmento residencial y un tanto obligada por la oferta comercial lanzada por Avantel, su competidor más cercano, anunció el lanzamiento de su servicio empaquetado denominado Masternet (Fig. 5.17), mismo que al igual que el de Avantel – NetVoice –, ofrece servicio de telefonía residencial IP y acceso a Internet de banda ancha de 256 y 512 kbps por una cuota mensual fija. Los servicios de telefonía IP incluidos en dichas soluciones pretenden contrarrestar la oferta de nuevos operadores como IUSACom, que ya ofrecían telefonía sobre Internet con llamadas locales y de larga distancia nacional a bajo costo e incluso, en algunos planes, de manera ilimitada.

Fig. 5.17 Servicio Masternet de Alestra

Maxcom

Maxcom por su parte, no se quedó atrás y anunció el lanzamiento oficial de su nuevo servicio Triple Play mediante el cual ofrecerá servicios de telefonía, Internet de banda ancha y televisión a través de una sola infraestructura, una sola factura, una sola marca y con un único punto de atención al cliente.

En la Tabla II se muestra un análisis competitivo entre las principales empresas que ofrecen el servicio de telefonía IP en México.

ANALISIS COMPETITIVO				
EMPRESA	AT&T	Avantel	IUSACom	MAXCOM telefonía
Servicio	masternet	NetVoice	VOX IP	*Triple Play*
Sector al que va dirigido	Residencial y PyMEs	Residencial	Residencial	Residencial
Telefonía:				
Local	✓	✓	✓	✓
LDN				
LDI				
Acceso a Internet:				
25 6 kbps	✓	✓	×	✓
512 kbps	✓	✓	×	✓
Servicio de Televisión	×	×	×	✓
Portabilidad	✓	×	×	×
Cobertura	4 ciudades: México, Toluca, Guadalajara y Monterrey	3 ciudades: Cd. de México, Guadalajara y Monterrey	18 ciudades	Únicamente en Querétaro

Tabla II. Análisis competitivo de los principales proveedores de telefonía IP

Consideramos importante realizar una comparación de las tarifas entre las el servicio VOXIP de la empresa IUSACom y el servicio Masternet de Alestra, tal como se muestra en la Tabla III.

TARIFAS					
EMPRESA	VOX IP				masternet
Servicio	IUSA Com				AT&T
PLAN	**Local**		**Nacional**		**Esquema Modular**
	256 kbps	**512 kbps**	**256 kbps**	**512 kbps**	
Renta Mensual Fija	$699.00	$849.00	$849.00	$977.00	$699.00
Local	Ilimitado	Ilimitado	Ilimitado	Ilimitado	Ilimitado
LDN	$1.25 el minuto	$1.25 el minuto	Incluye 600 mins Minuto ad.:$1.00	Incluye 600 mins Minuto ad: $1.00	$1.00 el minuto ó Ilimitado por $150.00
LDI	Por minuto:	Por minuto:	Por minuto:	Por minuto:	100 mins. por $200.00
EU y Canadá	$3.00	$3.00	$3.00	$3.00	Minuto adicional: $2.30
Centro América	$3.00	$3.00	$3.00	$3.00	$3.99
Sudamérica y Europa,África y Mediterráneo	$6.00	$6.00	$6.00	$6.00	$5.00
Asia y resto del Mundo	-	-	-	-	$9.99
Celular	Por minuto: $2.25	Por minuto: $2.25	Por minuto: $2.25	Por minuto: $2.25	100 mins. por $200.00 Minuto adicional: $2.25
Incremento de Velocidad (kbps)	✖	✖	✖	✖	De 256 kbps a 512 kbps $150.00
Activación	Sin costo	Sin costo	Sin coso	Sin costo	Sin costo

Tabla III. Tarifas de VoxIP y Masternet

No obstante que la presencia de servicios de voz y telefonía sobre IP en México es mucho menor a la que se tiene en países como Estados Unidos y la Unión Europea, de acuerdo con algunos de los principales fabricantes de equipos IP, entre ellos Siemens, Avaya, Cisco, Mitel y Nortel, México lleva el liderazgo en la región de América Latina.

Según Enrique Villegas, Director de Mercadotecnia de Avaya, la tendencia en el mercado mexicano es que la tecnología IP crezca por arriba de todo el sector de tecnologías de la información, pues se observa una expansión del 40% de estas nuevas soluciones.

Para el caso de telefonía IP, la consultora IDC estima que menos del 1% de las empresas en México tienen como sistema principal este servicio, no obstante, el 7.5% de este sector lo tiene como segunda opción o se encuentra en análisis para su implementación. De esta forma, en México existen amplias expectativas de

crecimiento de la telefonía IP. En los últimos meses se ha visto una mayor actividad comercial y de adopción promovida principalmente por Avantel a través de su servicio NetVoice, Alestra a través de Masternet y otras empresas como IusaCom con su servicio VOXIP.

Por su parte, Jorge Escribano, Director de Comunicación de Alestra, señala que hay una tendencia clara para migrar hacia la tecnología de VoIP. “En el futuro estos servicios serán provistos mediante acceso de Internet de banda ancha con calidad de teléfono a teléfono, no de computadora a teléfono o a computadora, ahorro en costos, seguridad y movilidad”. El funcionario destacó que: “El mercado residencial, que cuenta con acceso a Internet de banda ancha, va a buscar ahorros en larga distancia nacional e internacional”, en tanto que el mercado corporativo, “va a permanecer fiel a las operadoras que le garanticen un servicio telefónico con: excelencia en la operación; proximidad y eficiencia en el servicio post-venta, flexibilidad en la oferta que se adapte a las necesidades de cada segmento y mejoras del costo total de pertenencia”. Cita Montemayor, Director General de Marcatel,

Las grandes ventajas de las soluciones de telefonía IP residen en el hecho de ofrecer simultáneamente servicios de voz con fiabilidad, eficiencia en costo y alta velocidad de transmisión de datos. Para el segundo trimestre del 2006, Select estima que el 70% de las mega-empresas, 65% de los grandes corporativos, 60% de las organizaciones intermedias, 20% de las medianas y 20% de las pequeñas planean incorporar al menos una solución de voz sobre redes privadas de datos. Se estima que el mercado IP registre crecimientos anuales promedio de 40%, siguiendo claramente la tendencia mundial que para el 2007 hará que el 20% del tráfico de voz se dé a través de la tecnología tradicional y el restante 80% mediante IP.

En este sentido, analistas del sector afirman que en este y el próximo año marcarán el momento de migración de la telefonía analógica a la IP aprovechando las nuevas necesidades de las empresas – voz, datos e Internet bajo una sola conexión – así como el número de conexiones de banda ancha con que se cuenta.

En lo que se refiere al mercado global, Skype lidera el mercado actual en telefonía por Internet. Tiene más de 100 millones de usuarios registrados y más de 2 millones de clientes que pagan por servicios como buzón de voz y conexiones con la RTC publica. Existe además multitud de adaptadores y accesorios Skype para conectarse a la red inalámbrica, RTC pública y movil. Un ejemplo es el teléfono IP inalámbrico que permite realizar tanto llamadas convencionales como gratuitas a través de la red Skype. Dispone de una tecla que indica por pantalla los contactos activos en ese momento a Skype. La base que incorpora se conecta al puerto USB del PC (Fig. 5.18).

Fig. 5.18 Teléfono inalámbrico de Skype.

El servicio FWD es una aplicación abierta de VoInternet que facilita a los usuarios la comunicación con intercambio de voz, video, o texto, comunicaciones "peer-to-peer" entre usuarios FWD (Fig. 5.19). Los abonados usan un teléfono SIP o PC para hacer llamadas entre ellos sin usar la RTC pública. Los dispositivos de usuario son los que establecen la conexión y gestión de llamada.

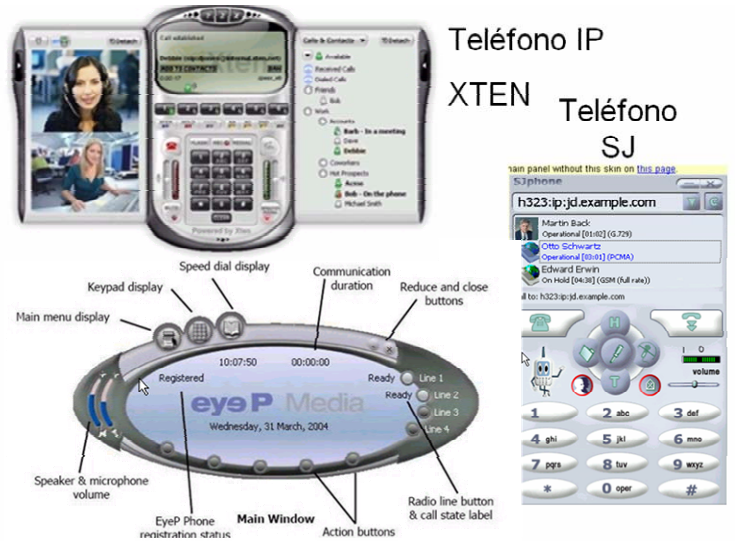


Fig. 5.19 Teléfonos del servicio FWD

5.2 ASPECTOS REGULATORIOS

La importancia real de la telefonía IP en el plano regulatorio es que puede llegar a ser un servicio sustitutivo de la telefonía vocal tradicional y, en consecuencia, su regulación ha de tener en cuenta el posible impacto comercial y de competencia respecto al servicio tradicional de voz.

Los enfoques regulatorios permiten establecer tres grupos generales de política nacional:

- Países que incluyen la telefonía IP dentro de su sistema de reglamentación, o que la toleran sin someter a reglamentación la telefonía IP.
- Algunos países en desarrollo no permiten la telefonía IP. Estas políticas suelen ser interesadas por los impuestos sobre estas tecnologías
- Administraciones donde la regulación es incierta, o esperan a decidir más adelante según la evolución del mercado. Las decisiones de regulación estan unidas a políticas de largo plazo en estructuras de telecomunicaciones

Por ejemplo, para el caso del servicio FWD, la FCC ha declarado (2/2004) que se considera como "servicio de información", no de "telecomunicaciones", considerando:

- Servicio ofrecido desde un servidor en Internet. No se proporciona transmisión directamente.
- Servicio de "directorio", o "traducción" con "indicador de presencia". Sin cargos ni cuotas.
- Cobertura del servicio, numero totalmente portátil.
- VoIP como agregado, no ofrecido directamente por FWD

Las consecuencias son que FWD y similares no están sujetos a la regulación de las telecomunicaciones.

Respecto al aspecto regulatorio de la oferta de estos servicios, la COFETEL ha establecido en su informe de labores de 2004 que: "los servicios públicos de telefonía básica únicamente pueden ser ofrecidos por aquellos que cuentan con una concesión para instalar, operar y explotar una red pública de telecomunicaciones que les autorice la prestación de dicho servicio, por lo que en tanto no exista una regulación específica para la oferta de voz sobre IP en el país, cualquier empresa interesada en ofrecer tales servicios deberá contar con una concesión y observar los ordenamiento jurídicos vigentes".

No obstante, varias empresas aprovechan los vacíos y huecos legales, así como la falta de eficiencia por parte de las autoridades, para ofrecer toda una gama de productos y servicios que permiten el transporte de voz a través de la red sin que ésta pase por la red de los operadores que están debidamente autorizados para hacerlo.

5.3 REPERCUSIONES DE ADOPTAR LA TECNOLOGÍA VOIP

Las consecuencias de la adopción de los servicios VoIP son profundas, ya que para consumidores y negocios implica ahorros sustantivos en sus gastos de telecomunicaciones, en tanto que para la industria implica un reordenamiento radical.

Sin embargo, los operadores tendrán que considerar ciertas particularidades en la provisión de sus servicios de transmisión de datos, como una mayor demanda de ancho de banda y conectividad, IP simétrico, así como calidad de servicio garantizada en cuanto a disponibilidad, latencia, ancho de banda y priorización de tráfico.

Por el lado de los clientes, de acuerdo con un estudio realizado recientemente por IDC, la principal razón que llevará a las empresas a migrar de sus soluciones de voz tradicionales a telefonía IP, será el ahorro en costos.

Lo anterior se debe a que la mayoría de las empresas de todos los tamaños se están viendo presionados a reducir sus inversiones de capital en telecomunicaciones y gastos operativos. Es por ello que la convergencia, es decir, la integración de voz y datos en una misma red, emerge como una solución real para la reducción de costos y aumentos en la productividad.

De esta forma, la consultora indicó que el 37% de las empresas han planeado implementar telefonía IP en los próximos años. Según el estudio, el 46% lo hizo en 2004, 28% lo hará en 2005 y el 4% en 2006 o posterior, en tanto que el 22% restante no sabe cuando implementará la solución.

5.4 TENDENCIAS DE LA TECNOLOGÍA VOIP

Las telecomunicaciones están experimentando uno de los procesos de cambio más intensos y decisivos que hasta ahora se conoce. Después de la desregulación, de la modernización y de las privatizaciones que se iniciaron en todo el mundo a mediados de los años 1980, se cayó en una crisis en el año 2000. Entre los años 2000 al 2003 esta crisis de las telecomunicaciones se caracterizó porque en las empresas del sector hubo escaso o nulo crecimiento (en comparación con la década anterior), hubo descenso en las inversiones y se dieron grandes quiebras y fraudes corporativos (MCIWorldCom, Global Crossing, etc) que afectaron las ganancias de todo el sector.

A partir de 2004, un nuevo factor de crisis comenzó a emerger para las grandes empresas de telecomunicaciones, en particular para aquellas que han sido principalmente operadoras telefónicas y paradójicamente este factor de crisis es la evolución tecnológica que está en marcha. Esta evolución cambiará definitivamente el destino y la estructura del sector de las telecomunicaciones en todo el mundo y, en consecuencia, cambiará el rumbo y la estructura

de todas las empresas del ramo. A esta evolución se le conoce genéricamente como las Redes de Próxima Generación (RPG) o Redes de Nueva Generación (RNG).

Desde hace algunos años era notoria la tendencia de un proceso de convergencia, es decir, que todos los servicios y redes se estaban integrando. Esta convergencia ha seguido evolucionando pues existe la plataforma tecnológica que permite la integración plena de todos los servicios de telecomunicaciones en una sola red. Esta plataforma tecnológica es la de los protocolos Internet o protocolos IP, que constituyen los cimientos de las redes de la próxima generación. La evolución tecnológica permite que empresas y servicios de telecomunicaciones que antes trabajaban de manera separada, (por ejemplo empresas de servicios de voz, de servicios de televisión por cable y de servicios de datos, etc.) puedan ya, en términos tecnológicos, unificarse y proporcionar todos estos servicios a la vez, usando sus propias redes bajo el protocolo Internet. Podemos considerar la convergencia bajo distintos aspectos tales como redes, servicios y regulación. Se están ofreciendo gradualmente diferentes servicios en convergencia que contribuyen a aumentar la oferta disponible al público. La idea además es tener un solo dispositivo en el que se puedan integrar todos los servicios tal como se muestra en la Fig. 5.20.

Fig. 5.20 Convergencia e integración de servicios

Una RNG es una red capaz de transportar y proveer todos los servicios de telecomunicaciones mediante la convergencia de protocolos, redes, normas y tecnologías que ya existen, de otros que se están configurando y de otros más que estarán disponibles en el futuro.

Es importante tener claro que se está hablando de una evolución tecnológica de largo alcance en telecomunicaciones en la que la Voz Sobre Protocolo Internet o VoIP constituye tan sólo una primera etapa.

Hasta el momento, es posible ubicar dos grandes periodos en el desarrollo de la Redes de Próxima
Generación.

Del año 2005 al 2010, en el que la voz sobre IP y, en general, los protocolos Internet o protocolos IP y los servicios de banda ancha se volverán dominantes y determinantes.

La dominancia de los protocolos IP abre la posibilidad de una nueva fase de convergencia en una red única de telecomunicaciones, caracterizada aún por las “redes tontas” (que sólo transmiten) y por las “terminales inteligentes” (que decodifican y hacen el trabajo detallado).

En la medida en que las redes IP tengan una arquitectura distinta a la de las redes telefónicas tradicionales, entre el 2005 y el 2010, habrá cambios sustanciales en la

arquitectura de las redes. Por ejemplo, desaparecerán progresivamente las centrales de conmutación y en general, la Red de
Telefonía Pública Conmutada (RTPC) evolucionará y se transformará radicalmente.

Segmentos de negocios como la larga distancia, el servicio medido y otros dejarán de ser rentables e irán disminuyendo progresivamente o desaparecerán. De ser una parte sustancial en los ingresos de las empresas telefónicas tradicionales, la larga distancia se volverá un servicio que las empresas no extinguirán de inmediato pero que algunas conservarán, muy reducido, para unos cuantos usuarios. El servicio medido también desaparecerá como fuente de ingresos y será sustituido por tarifas planas.

Las empresas de telecomunicaciones competirán por servicios de valor agregado cada vez más complejos y personalizados para sus clientes. Se anticipa, de hecho, una competencia nueva, cada vez más intensa y agresiva entre las empresas de telecomunicaciones. Previsiblemente, la competencia más fuerte será entre las telefónicas en evolución, con las empresas de cable y las nuevas empresas de VoIP. También podrían sumarse a esta competencia empresas eléctricas, en la medida en que también entre el 2005 y el 2010 hayan avances considerables en la convergencia de las redes de telecomunicaciones y las redes eléctricas (en principio mediante la tecnología PLC o comunicación por línea de potencia) configurándose la estructura de una red universal.

Del año 2010 en adelante, los expertos pronostican una segunda etapa en el desarrollo de las RPG, en la que muy posiblemente habrá no sólo “terminales inteligentes”, sino “redes inteligentes” que abren un universo inimaginable de posibilidades para servicios de telecomunicaciones cada vez más complejos y de cada vez más alto valor agregado.

El valor de la RTPC existente puede quedar reducido a cero, a menos que esta red evolucione rápida, eficiente y competitivamente para convertirse en una red de próxima generación. Este es uno de los desafíos más importantes que enfrentan las empresas telefónicas tradicionales, cuya red, desde el punto de vista de los clientes, se convertirá en una de varias para acceder a los nuevos servicios y, eventualmente, a todos los servicios de telecomunicaciones.

La transmisión VoIP reemplaza las centrales de conmutación telefónica; en lugar de usar centrales telefónicas convencionales, los paquetes “de voz” son transportados como datos. Se pasa del concepto de “conmutación por circuitos” al de “conmutación por paquetes”.

La evolución tecnológica en telecomunicaciones ocurre en torno de dos factores principales: Servicios y tecnologías de Acceso al Cliente. El primero incluye voz, datos, video, redes, juegos, etc., es decir todo aquello que es “protocolizable” a IP y banda ancha, encabezados por VoIP. El segundo factor incluye ADSL, Cable, WiFi, WiMax, PLC y satélite, entre otros.

La telefonía tradicional tiene que insertarse en estos nuevos esquemas de competencia tecnológica y de mercado. Convivirá con las nuevas tecnologías, que irán bajando gradualmente sus costos hasta hacerse tanto o más competitivas que la vieja tecnología y que, en consecuencia, la irán desplazando.

VoIP hace peligrar los ingresos de las compañías telefónicas tradicionales que se encuentran todavía ofreciendo fundamentalmente servicios de voz. Muchas de las empresas principales ya disponen de tecnologías VoIP o se preparan para ello. Lo cierto es que el simple negocio de la voz está dejando y dejará de ser negocio. En algunos países, existe además una tendencia decreciente de abonados en líneas fijas, que se presenta como irreversible. En Japón ha pasado de 62.85 millones (pico histórico) en 1998, a 60.77 millones en marzo de 2003, y sigue a la baja. El tráfico en las líneas fijas pasó de 5,580 millones de horas en el 2000, a 3,990 millones de horas en 2002.

Desde 1995, en el grupo corporativo NTT de Japón no se generó ningún puesto de trabajo nuevo ni vacantes en el segmento de telefonía fija. British Telecom está estimulando el desarrollo de teléfonos inalámbricos VoIP que se conectan a redes públicas en Londres, lo que ocasionaría una reducción de hasta 70% de las ganancias en las compañías celulares.

En Estados Unidos VoIP es ofrecida ya por:

- Verizon, SBC, Bell South y Qwest, a través de ADSL
- Compañías de cable, a través de cable modem
- AT&T
- Compañías específicas de VoIP, como Vonage en New Jersey, con más de 600,000 suscriptores en poco más de dos años, Net 2 Phone y 8 x 8
- Universidades, empresas y ciudades, sobre redes privadas

Vonage, el operador más grande de VoIP en Estados Unidos, rebasó los 600,000 suscriptores en abril de 2005 y ha crecido a un ritmo de 15,000 nuevos clientes por semana. Vonage está ofreciendo números mexicanos a muchos de sus clientes. Estos números cuestan al cliente 5 dólares mensuales, y permiten que todas las llamadas que les hagan a ese número desde México tengan costo de llamada local.

Lo mismo está haciendo otra empresa, Lingo, que ofrece números telefónicos de la Ciudad de México, Guadalajara y Monterrey para sus clientes en Estados Unidos. Vonage y Lingo han comprado los números a una empresa telefónica mexicana.

En Estados Unidos, la empresa Vonage está ofreciendo actualmente cuatro esquemas de tarifas: la más baja ofrece, por 15 dólares mensuales, 500 minutos de llamadas a cualquier lugar de Estados Unidos y Canadá. La tarifa ilimitada de Vonage es de 25 dólares mensuales.

Para proveedores como Cisco Systems, Equant y Fonet Global, la telefonía basada en IP optimiza los procesos de comunicaciones de una empresa, porque permite hacer llamadas a través de una sola red para conectar sucursales, sin los grandes gastos de las llamadas de larga distancia.

Con o sin infraestructura de red propia, o con una infraestructura mínima, las nuevas empresas de VoIP, junto con las empresas de cable, están comenzando a competir por los clientes que ya disponen o que pueden tener acceso a conexiones de banda ancha. La plataforma tecnológica IP es prácticamente la misma.

Hay que señalar que en Estados Unidos las empresas de cable le llevan ventaja a las telefónicas en la expansión del servicio VoIP. En todo el mundo, las empresas de cable han gozado de privilegios regulatorios. No existe para ellas regulación de tarifas, ni compromisos de infraestructura y de calidad.

Cablevisión, el mayor proveedor de TV por cable en México, calcula llegar a 400,000 suscriptores en TV y a 50,000 en Internet de banda ancha al cierre de 2005. Puede entrar al negocio VoIP por sí sola, si logra la autorización, o bien en alianza con una empresa telefónica, comprando una concesión.

La evolución tecnológica de la red se complementa con la evolución operativa y de mercado de las telecomunicaciones, que en conjunto anticipan un panorama completamente novedoso dentro del sector de las telecomunicaciones, en comparación con lo que hoy aún predomina y con lo que ha predominado en los últimos años.

Frente a los nuevos escenarios de competencia y evolución tecnológica, muchas empresas están adoptando estrategias agresivas de reestructuración. En Estados Unidos, por ejemplo, la empresa Verizon ha anunciado despidos y espera cerrar cuatro de cada cinco centros operativos en los próximos años. Al igual que NTT en Japón, Verizon detuvo ya la actualización y el mantenimiento de los conmutadores más antiguos, lo que implica reducción de personal en estas áreas.

Empresas telefónicas de origen, como Verizon y SBC en Estados Unidos, que en conjunto operan más de 100 millones de líneas telefónicas tradicionales, están buscando incursionar en el mercado de la televisión, desplegando programas intensivos de reconversión de la red con fibra al hogar y conexiones de banda ancha. La estrategia de estas empresas es clara: evolucionar la red telefónica tradicional a una red de próxima generación de alta tecnología, para retener a los clientes que ya tienen y entrar de lleno en la competencia por nuevos clientes, en la medida en que se generalicen y abaraten los servicios de banda ancha. La operación de contenidos y la cautivación del cliente se están convirtiendo en estrategias corporativas para enfrentar la evolución de las telecomunicaciones

Por otra parte, la tecnología VoIP no está regulada. Las empresas de VoIP tienen esta ventaja competitiva adicional, sobre todo en comparación con las empresas telefónicas que enfrentan serios obstáculos regulatorios, por ejemplo para brindar servicios de “triple play” o de “cuadru-play” (triple play más móvil)

La expansión de una VoIP desregulada puede volver más inequitativo el mercado de telecomunicaciones en todo el mundo. Sin una regulación y sin políticas nacionales adecuadas, el servicio universal será cada vez más inaccesible.

Uno de los grandes retos en telecomunicaciones es la regulación, no sólo para los sindicatos sino para las sociedades (importancia de alianzas y coaliciones sociales). La competencia en el sector ya no será entre operadores, sino entre REDES. La telefonía fija ha sido la base de la regulación, y hasta ahora todos los modelos regulatorios han fracasado, por lo que es necesario plantear nuevos enfoques.

La competencia será, en efecto, entre redes, por más y mejores servicios de telecomunicaciones, en entornos de tarifas planas y de marcos regulatorios poco amigables para las empresas que vienen de ser operadoras telefónicas. Las telefónicas enfrentan también nuevos competidores provenientes de otros ámbitos tecnológicos que antes no figuraban en la prestación de servicios de comunicación, como es el sector de la tecnología en cómputo y de redes de transmisión de datos.

En cuanto a tecnologías de acceso al cliente, otra evolución tecnológica a la que nos enfrentamos es PLC (Power Line Communication) , que es la tecnología que permite la transmisión de datos por el segmento de baja tensión (o red secundaria) de las redes eléctricas, desde la subestación eléctrica al domicilio u oficina del cliente. La tecnología PLC ya está siendo probada e implantada en México.

En los nuevos escenarios de las RPG, los clientes podrán elegir entre diversas opciones para conectarse y disponer de los mismos servicios, sin distinguir si su compañía es telefónica, de TV por cable, de VoIP u otra. La diferencia estará dada por la diversidad de los servicios que éstas ofrezcan a los clientes, las tarifas y, en gran medida, por la calidad del servicio. Telmex utiliza ya la
tecnología VoIP para operaciones internas.

La creciente internacionalización de Telmex y las ventajas que la tecnología le ofrecen para esto, son también tema de debate y de definiciones para el Sindicato. Recientemente, Telmex contrató una línea de interconexión a lo largo de todo el continente a Global

Crossing, lo que permitirá reducir significativamente costos de interconexión, mandando todo el tráfico de llamadas desde Sudamérica directamente a México para de aquí conectarlas a Estados Unidos, el destino de 80% de llamadas de la región.

En resumen, VoIP anticipa una nueva era de las telecomunicaciones, etapa que se conoce ya como la de las “redes de próxima generación”. Las empresas de telecomunicaciones están experimentando y experimentarán cambios radicales. La evolución de las telecomunicaciones es una realidad con plazos, efectos y consecuencias predecibles. La evolución de la red es uno de los procesos más importantes del cambio en las telecomunicaciones.

CONCLUSIONES

La telefonía IP se basa en tomar los canales de voz convencionales, procesarlos y generar paquetes IP, recordando que un canal de voz convencional (TDM) implica un ancho de banda de 64 kbps surgido del muestreo de la señal de voz con una frecuencia de 8 kHz y el empleo de un conversor A/D de 8 bits, entonces los paquetes IP se transportan por una red IP hasta su destino final que puede ser un teléfono IP o bien un teléfono convencional, que en pocas palabras resume el sistema utilizado en esta tesis.

La adaptación entre los canales telefónicos convencionales y la red IP fue llevada adelante por Media Gateways, que para nuestro caso fue el Gateway SIP en el que se implantó la aplicación de SIP.

La aplicación desarrollada se basa en las especificaciones del protocolo SIP, cabe señalar que el RFC 2543 de este protocolo indica las características de cada uno de los campos que la componen. En cada uno de los campos del protocolo, se indica con exactitud los caracteres y los enunciados que deben ser enviados, para que se pueda establecer la sesión multimedia. Es de vital importancia que todos los equipos que utilicen el protocolo cumplan con cada uno de dichos caracteres, puesto que el contenido en bytes deberá corresponder entre los equipos que se comuniquen utilizando este protocolo, y en general cualquier protocolo. Si los campos que se envían entre dos equipos que se supone se comunican utilizando el mismo protocolo, no son iguales; entonces cada una de las terminales al intentar identificar los bytes que recibe, no podrá determinar a qué tipo de mensajes corresponden o el tipo de campo que ha recibido. Si la identificación entre los equipos terminales que se comunican entre sí, es errónea, no se podrá establecer una sesión multimedia en el caso del protocolo SIP, ni siquiera podrán identificar qué tipo de datos recibe.

Dado lo anterior, es necesario seguir una regla específica, como lo marca el RFC 2543 del protocolo SIP. Si las terminales siguen cada una de las estructuras de los campos tal y como lo dice el RFC 2543, es más probable que las terminales pueden establecer una comunicación exitosa. El problema que suele presentarse, es que algunos de los equipos terminales, no siguen del todo el protocolo y dificultan la comunicación. Existen distintos casos de modificación del protocolo:

- Cuando se agregan o suprimen campos al protocolo, no identificados previamente por las partes a comunicarse. Algunos de los campos que se envían en el cuerpo del mensaje no son obligatorios y por lo tanto las terminales pueden aún identificar el tipo de protocolo.
- Cuando se agregan o se suprimen datos o caracteres especiales, que sirven para identificación de contenido en protocolos entre terminales específicas.
- Cuando el tamaño, tanto del cuerpo del mensaje y del SDP, difieren del esperado en bytes, en muchas ocasiones una de las terminales no logra identificar el tipo de mensaje que está recibiendo. Por ejemplo: no sabe que el mensaje que ha recibido se trata del protocolo SIP.

Nuestra aplicación tiene cargado un tipo de protocolo SIP que tiene como base el RFC 2543 pero que tiene diferencias en cuanto a:

- Tamaño en bytes del SDP y del contenido del mensaje.
- Se suprimen campos en SDP como: a=rtpmap, a=ftpmap entre otros.
- Se agregan datos de identificación entre las terminales como: la cuenta de ingreso al servidor Yaltel, el número telefónico destino y las direcciones IP específicas destino – origen.

La principal diferencia de nuestra aplicación con respecto al RFC, es que se envía el teléfono destino de la llamada junto con la dirección IP del servidor SIP, puesto que normalmente la comunicación entre las terminales que se comunican vía SIP lo hacen utilizando las direcciones IP privadas o públicas asignadas dentro de su red.

Para lograr enviar todos los requerimientos que la empresa Yaltel solicitó, las características propias de la aplicación Voz sobre IP, las protecciones antifraude, etc. fue necesario utilizar un analizador de protocolos, que extrajera cada uno de los campos de las peticiones SIP y de las respuestas SIP que el servidor utilizaría para comunicarse con la tarjeta de voz; éstos mensajes fueron nuestra referencia para tratar de igualarlos y lograr que el servidor SIP entendiera los mensajes generados por nuestra aplicación, de ahí la importancia de que los mensajes entre el servidor SIP y la tarjeta de voz sean iguales, de ahí la importancia de mencionar que aunque existe una RFC para el protocolo SIP, existen modificaciones al protocolo que permiten aún así establecer una sesión multimedia.

Por lo tanto, es muy útil saber modificar un protocolo con las características particulares que los equipos terminales necesiten para establecer un tipo de sesión. Si dos terminales utilizan el protocolo SIP, podrán hacer las modificaciones pertinentes para mantener las bases del RFC y seguir comunicándose entre sí con éxito. Una de las opciones más efectivas para igualar las peticiones y respuestas SIP es utilizar, como ya se mencionó un analizador de protocolos *Etherreal*, y un software de comparación de cadenas de datos *Compare it*.

Una vez hecha la aplicación característica al tipo de sesión que se establecerá, es muy útil saber o determinar si la comunicación exitosa entre la tarjeta de voz y el servidor SIP se podrá llevar a cabo con cualquier otro servidor en la red IP, es decir, qué tan portable es la tarjeta con la aplicación desarrollada; y la respuesta es No es del todo portable, debido a que las peticiones SIP y las respuestas enviadas están conformadas de tal manera que si alguno de los parámetros en el cuerpo del mensaje o del SDP cambia, entonces será necesaria modificar la aplicación. Por ejemplo cuando sea necesario cambiar las direcciones IP destino-origen, la cuenta de ingreso al servidor SIP, etc.

Una de las soluciones a este problema es realizar una ventana de configuración para los parámetros internos de la tarjeta, en el que el usuario de la tarjeta pueda modificar en cualquier momento dichos parámetros.

La principal ventana de mantenimiento para nuestra aplicación consistió en una hyperterminal, con la que se podía monitorear su desempeño en conjunto. El éxito de la aplicación proviene también, de la programación del código por procesos en paralelo sobre la tarjeta, ejecutados cada vez que cada uno de éstos se requiera. Si la programación no se hace por procesos en paralelo, entonces podremos consumir la capacidad de procesamiento de la tarjeta y el tiempo de respuesta a las peticiones de cada una de las funciones será largo, debido a que no se podrá interrumpir el procesamiento de una tarea hasta que ésta finalice.

Es muy útil mantener una secuencia de impresiones en la pantalla de la hyperterminal, pues de esta manera se podrá identificar el punto del código en general en la que surgió un problema durante el lazo infinito del ciclo RUN. Pero es mucho más recomendable crear un archivo de impresión de errores (log) que sirva de referencia para consultar cualquier tipo de problema en la ejecución del código o simplemente para confirmar su correcto funcionamiento.

Al poder tener procesos en paralelo o jerarquizados, la aplicación es capaz de transitar de una función a otra dependiendo de su grado de prioridad y ésta a su vez podrá ser interrumpida por otra de mayor jerarquía. La aplicación está desarrollada de tal manera que se han jerarquizado los procesos y así se pueda mantener una secuencia normal de establecimiento de llamada o de terminación de llamada, por ejemplo: si la aplicación se encuentra en un estado de espera, puede ser interrumpida por el proceso de levantar el auricular y ésta a su vez puede ser interrumpida por la marcación de un dígito, posteriormente de este estado puede pasar al envío de la petición SIP. Cabe señalar que la jerarquización de la aplicación permitirá que cuando uno de los procesos se encuentre en espera de un estado siguiente, la aplicación en general podrá permanecer en un lazo principal de ejecución (estado RUN), el cual podrá ser interrumpido bajo cualquier proceso.

Para lograr establecer la transferencia de datos entre el host (192.168.1.189) y la tarjeta de voz (192.168.1.190), es necesario que exista una definición explícita de los dos puntos de comunicación, que en este caso sería definir los tipos de sockets y los nombres de los sockets. Si la definición de los sockets no es la adecuada entonces, jamás se podrá intercambiar información entre los puntos. En nuestra aplicación el socket necesario para el protocolo SIP es uno del tipo UDP (debido a que necesitamos enlazar rápidamente los dos puntos y en tiempo real) UDP nos permite hacer este tipo de transmisión. Además los dos puntos deben estar comunicándose al mismo número de puerto abierto para el socket, en el caso SIP el puerto 5060, puesto que existen un gran número de puertos de comunicación, pero la RFC indica que el puerto 5060 deberá ser usado para este caso.

Es muy importante mencionar que para poder recibir mensajes SIP dentro de una red local, primero se debe configurar el router de la red o en su caso el firewall, para que permita pasar los mensajes hacia su destino final; indicándole que todo el tráfico que vaya al puerto SIP o el puerto de voz de la tarjeta de red o el servidor SIP logre llegar sin rechazo al destino final, es decir, se abran los puertos al paso del tráfico.

En cuanto a la calidad de la voz, es importante seleccionar un tipo de códec de voz para que ésta no se vea afectada por los medios de transmisión y se pueda eficientar el ancho de banda. El códec que utilizamos fue el G.711 debido a que, en primera instancia, no necesitamos pagar una licencia por uso y la tasa de transmisión es adecuada para el transporte de la voz. La tarjeta de voz, tiene cargada por default una serie de códecs que pueden ser seleccionables en caso de que se requiera mejorar la calidad de transporte de voz.

Con nuestra experiencia, podemos indicar que el códec G.711 cumple con la calidad necesaria para un buen servicio telefónico, pero si se desea eficientar el proceso, entonces podemos sugerir el códec G.729 ya que con éste se maneja mejor el ancho de banda y se puede conseguir una mejor transmisión de voz, el único inconveniente con este códec es que es necesario pagar por el uso del mismo y que aunque efectivamente se eficienta el ancho de banda, aumenta el tiempo de retraso de la transmisión de la voz y disminuye la calidad en general de la voz. Así que seguramente un códec nos ayudará a utilizar mejor el ancho de banda del canal de transmisión reduciendo la tasa de transmisión en gran cantidad desde 64 Kbps para transmisiones TDM hasta cerca de 5-6 Kbps, pero entre más baja sea la tasa de transmisón el delay aumentará y para las transmisiones de voz es de vital importancia que éstas sean en tiempo real, además como ya se dijo entre más sea la compresión se perderá más información para recuperar la voz con una buena calidad en el receptor.

Este trabajo de tesis no tiene los alcances para poder comparar la eficiencia de ambos códecs, ya que no se consiguió obtener una licencia de uso, además en la aplicación solicitada no se requería utilizar de primera instancia el códec G.729.

Un aspecto que tenemos que señalar, es que la calidad del audio en el auricular del teléfono público depende en gran medida de la conexión con la tarjeta de voz, pues dado que al establecerse la llamada el teléfono cambia de polaridad ± 48 V, entonces la voz se escucha atenuada. La recomendación que hacemos es que la tarjeta debe ir acompañada en el gateway final por un amplificador de la banda de voz (300 Hz – 4 kHz) que soporte el cambio de polaridad o en su caso que realice el cambio de polaridad.

Las pruebas realizadas nos confirmaron que si el gateway se conecta a Internet marcando a un módem y utilizando la conexión dial-up con tasas de transmisión de menos de 56 kbps, la transmisión se verá severamente afectada, pus se producirá retraso de la voz en la conexión y por supuesto disminuirá su calidad. Es mucho más recomendable contar con una conexión de banda ancha de por lo menos 256 Kbps, que puede ser Asimétrica ADSL, y partiendo del hecho de que en una transmisión E1 se cuentan con 32 time slots de 64 Kbps cada uno, que en conjunto nos da una tasa de 2.048 Mbps, podemos entender que cuando utilizamos solo 64 Kpbs para la transmisón de la voz, solo estamos utilizando 1 time slot de los 32 y por lo tanto para que un time

slot específico arribe a su origen, tendrá que esperar el paso de 31 time slot restantes haciendo así que la transmisión se retrase. Por lo tanto, si utilizamos más time slots el retraso será menor y podremos enviar más bits de la voz en cada ráfaga. En el caso de la tasa de 256 Kbps, estamos utilizando 3 time slots.

El proyecto del gateway SIP, supone que todos los teléfonos públicos se conectarán directamente a los pots telefónicos de donde tomarán el servicio. Las particularidades del proyecto son:

1. La distancia de los teléfonos públicos al gateway puede ser considerable o no, pero si lo es, entonces la señal de audio tendrá una mayor atenuación comparable con distancias menores.
2. El gateway deberá contar con una conexión de banda ancha obligatoria.
3. Alimentación de 5 V DC siempre activa con corriente de por lo menos 3 A. Recordemos que por el número de teléfonos que se conectarán, se demandará más corriente al gateway.
4. Los procesos de todos los teléfonos conectados dependen del procesador del gateway, por lo que si éste presenta algún tipo de problema, todos los teléfonos dejarán de tener servicio.
5. Si la conexión telefónica sufre algún tipo de error de software es probable que el procesamiento en la conexión de los demás teléfonos también presente errores.
6. No existe redundancia de procesamiento y no se cuenta con un archivo específico de log para errores.
7. El host del gateway, que en nuestro caso fue la máquina 192.168.1.189, utiliza la transmisión serial FTP para comunicarse con la tarjeta de voz y realizar actualizaciones o cambios en el funcionamiento de las aplicaciones que realizará el gateway junto con cada uno de los teléfonos públicos conectados, pero el host tendrá que estar ubicado a unos metros del gateway o en su caso se tendrá que conectar el host al gateway cada vez que se requiera.
8. La enorme ventaja del proyecto es que se aprovechan los recursos de la Red de Datos y no de la red Telefónica Pública, por lo que tendremos comunicación de voz mundial a costos menores que los convencionales.

GLOSARIO

A .....

ACK (Acknowledgement): Mensaje de reconocimiento para señalar que la información ha sido recibida correctamente.
ADSL (Asymmetric Digital Subscriber Line): Tecnología que permite transmitir por cables de cobre una señal cuya velocidad es más alta de entrada al usuario que de salida.
Algoritmo: Regla o proceso bien definido para llegar a la solución de un problema. En el entorno de las redes, con los algoritmos se determina la mejor ruta para enviar el tráfico desde el origen hasta un destino particular.
API: Interfase de Programación de la Aplicación. Especificación de las convenciones para llamar funciones, que define una interfase hacia un servicio.
ASCII: Código Estándar Americano para el Intercambio de Información. Es un código de 8 bits para la representación de caracteres (7 bits más paridad).

B .....

Banda Ancha: Sistema de transmisión que multiplexa muchas señales independientes a través de un cable. En la terminología de las telecomunicaciones, significa cualquier canal que tenga un ancho de banda mayor que el de un canal de voz (4 kHz).
Banda Base: Característica de una tecnología de redes donde sólo se utiliza una frecuencia portadora.
Baud: Unidad de velocidad de señalización igual a la cantidad de elementos discretos de señal transmitidos por segundo.
Bit (Binary Digit): Dígito binario utilizado en el sistema de numeración binario. Puede ser 0 o 1.
Bps (Bits per second): Unidad de medida de la velocidad de datos digitales. Usualmente se utilizan múltiplos como kbps, Mbps, Gbps.
Buffer: Área de almacenamiento utilizada para el manejo de datos en tránsito.
BW (Bandwidth): Diferencia entre la frecuencia más alta y la más baja disponible en una red. Este término también se utiliza para describir la tasa máxima de transmisión con un medio de transmisión o un protocolo determinado.

C .....

Canal: Trayectoria de comunicación.
CAS (Channel Asociated Signalling): Método de señalización utilizado en los circuitos digitales donde la señalización se transmite junto a la señal de voz. Es reemplazado por SS7.
CCS (Common Channel Signalling): Señalización por Canal Común. Sistema de señalización utilizado en redes telefónicas, que separa la información de señalización de la información del usuario. Se asigna un canal específico exclusivamente para transportar la información de señalización de todos los demás canales del sistema.
CODEC: Codificador Decodificador. Dispositivo que suele utilizar la técnica PCM para transformar señales analógicas en un flujo de bits digitales y las señales digitales en señales analógicas.
Compilador: Un compilador acepta programas escritos en un lenguaje de alto nivel y los traduce a otro lenguaje, generando un programa equivalente independiente, que puede ejecutarse tantas veces como se quiera. Este proceso de traducción se conoce como compilación.
Conector RJ: Son conectores estándar que originalmente se usaban para conectar líneas telefónicas. Los conectores RJ se utilizan ahora para conexiones telefónicas y para redes 10BaseT y otro tipo de conexiones de red. Los conectores RJ-11, RJ-12 Y RJ-45 son tipos de conectores RJ muy populares.
CPU (Central Processing Unit): Unidad de procesamiento central de una computadora.

D

Dirección IP: Es una dirección de 32 bits asignada a los anfitriones que utilizan TCP/ IP. Una dirección IP pertence a una de las cinco clases (A, B, C, D y E) y se escribe como 4 bytes separados con puntos (formato decimal punteado).
Dirección MAC: Es la dirección estándar de la capa de enlace de datos que se requiere para cada puerto o dispositivo que se conecta a una LAN. Las direcciones MAC tienen una longitud de 6 bytes y están controladas por IEEE.
DNS (Domain Name System): Sistema de Nombramiento de Dominios. Sistema utilizado en Internet para traducir los nombres de los nodos de red en direcciones.
DTMF (Dual Tone Multi-Frequency): Tono Doble de Multifrecuencias. Es el uso de dos tonos simultáneos en la banda de voz para el marcado telefónico.

E

E1 (European 1): Denominación comercial de un circuito de 2048 kbps, conformado por 32 time slots de 64 kbps cada uno.
Enlace (link): Canal de comunicaciones de red que consta de un circuito o trayectoria de transmisión y todo el equipo asociado entre el emisor y el receptor.
Ethernet: Especificación de LAN de banda base. Las redes Ethernet utilizan el método de acceso CSMA/CD y corren sobre una gran variedad de tipos de cables de 10 Mbps. La red Ethernet es similar a los estándares de la serie IEEE 802.3.
ETSI (European Telecom Standard Institute): Instituto para el ámbito de Europa que normaliza las telecomunicaciones. Similar a la ITU-T.

F

FastEthernet: Basado en la norma Ethernet, permite la interconexión a 100 Mbps mediante el uso de cables de cobre (en una LAN) o fibra óptica.
FTP (File Transfer Protocol): Protocolo definido en el ámbito de Internet para permitir la transferencia de archivos entre terminales.
FXS: Puerto que simula la central telefónica.
FXO: Puerto que simula un aparato telefónico.
Firewall: Ruteador o servidor de acceso, designados como un búfer entre cualesquiera redes públicas conectadas y una red privada. Utiliza listas de acceso y otros métodos para asegurar la confiabilidad de la red privada.

G

Gateway: Es un servidor, que proporciona a clientes conectividad hacia el mundo exterior, estén o no dentro de una red privada.
GigabitEthernet: Red de datos que permite la interconexión de redes LAN utilizando el mismo protocolo Ethernet pero a la velocidad de 1000 Mbps por fibras ópticas.

H

Hetz: Medida de frecuencia, abreviada Hz.
HFC (Hybrid Fiber/Coax): Arquitectura de red que utiliza una combinación de fibra óptica y cable coaxial, lo que permite una mayor capacidad de canales, mayor nivel de señal y mejora la confiabilidad.
Host: Sistema de computación en una red. Es similar al término nodo excepto en que el host por lo común implica un sistema de computadoras.
HTTP (HyperText Transfer Protocol): Protocolo de transferencia utilizado para el servicio Web (www) en Internet. El lenguaje de escritura es el HTML.
Hyperterminal: Ventana de comunicación para datos vía puerto serial, entre el sistema operativo Windows de una computadora y otro dispositivo. En esta ventana se pueden observar todos los datos que se están recibiendo y enviando entre ambos dispositivos.

I

IEEE (Institute of Electrical and Electronic Engineers): Instituto de Ingenieros en Electrónica y Electricidad. Organización profesional cuyas actividades incluyen el desarrollo de los estándares de comunicaciones y de redes.
IEEE 802.x: Grupo de normas para redes: 802.3 para Ethernet, 802.5 para Token Ring, 802.6 para MAN, etc.
Internet: Término que se refiere a la red global más grande del mundo, la cual conecta a decenas de miles de redes en todo el mundo.
IP (Internet Protocol): Protocolo de Internet. Protocolo de la capa de red en la pila TCP/IP que ofrece un servicio de red sin conexión. El protocolo IP proporciona características de direccionamiento, especificación del tipo de servicio, fragmentación y reensamblado y seguridad.
ITU-T (International Telecommunication Union): Unión Internacional de Telecomunicaciones, sector de estandarización de las telecomunicaciones. Organismo internacional que desarrolla estándares para las diferentes tecnologías de telecomunicaciones a nivel mundial.

L

LAN (Local Area Network): Red de datos de alta velocidad y baja tasa de errores, que cubre un área geográfica relativamente pequeña (hasta de algunos miles de metros). Las LAN´s conectan estaciones de trabajo, periféricos, terminales y otros dispositivos en un solo edificio u otra área geográfica limitada.
Log (Login/Logon): Procedimiento por el cual se inicia una sesión para el acceso a un sistema. El procedimiento de denomina *Logout*.

M

MAC (Medium Access Control): Protocolo de capa 2, el cual sirve para identificación de los componentes en una LAN. Las direcciones MAC constan de 6 bytes.
MAN (Metropolitan Area Network): Es una red que cubre un área equivalente a una ciudad. Puede realizarse mediante IEEE 802.6, FDDI, GigabitEthernet, etc.
Máscara de subred: Máscara de dirección de 32 bits para indicar los bits de una dirección IP que se están utilizando para las direcciones de la subred.
Modem (Modulator-Demodulator): Modulador Demodulador. Es un dispositivo que convierte señales digitales y analógicas. En el nodo origen, un módem convierte las señales digitales en una forma apropiada para su transmisión a través de dispositivos de comunicación analógica. Los módems permiten la transmisión de datos a través de líneas telefónicas de voz.

N

NAT (Network Address Translation): La Traducción de Direcciones de Red, o NAT, es un sistema que se utiliza para asignar una red completa (o varias redes) a una sola dirección IP. NAT es necesario cuando la cantidad de direcciones IP que nos haya asignado nuestro proveedor de Internet sea inferior a la cantidad de equipos de cómputo que queramos que accedan a Internet.
NNTP (Network News Transport Protocol): Es un Protocolo de red basado en tiras de textos envíados sobre canales TCP de 7 bit ASCII . Es usado para subir y bajar así como para transferir artículos entre servidores.
Nodo: Punto terminal de una conexión de red o unión común de dos o más líneas en una red. Los nodos pueden ser procesadores, controladores o estaciones de trabajo.

O

OSI (Open Systems Interconnect): Es el programa de estandarización internacional creado por la ISO y la ITU-T para desarrollar estándares para las redes de datos que faciliten la interoperabilidad de equipos fabricados por diferentes proveedores.

P

PBX (Private Branch Exchange): Central telefónica privada, es un conmutador telefónico, analógico o digital, ubicado en las instalaciones del suscriptor y que se utiliza para conectar redes telefónicas públicas y privadas.
PCM (Pulse Code Modulation): Transmisión de información analógica en formato digital a través del muestreo y codificación de muestras con un número fijo de bits.
POP (Point of Presence): Se refiere al punto donde un operador de Internet ISP tiene disponible el acceso de los usuarios.
POT: Punto de conexión de teléfonos estándar con conectores RJ-11.
Protocolo: Descripción formal de un conjunto de reglas y convenciones que rigen el modo de los dispositivos de una red de intercambiar información.
PSTN (Public Switched Telephone Network): Denominación genérica para las redes de telefonía pública convencionales.
Puerto: Interfase en un dispositivo de interconectividad de redes.

Q

QoS (Quality of Service): Calidad de Servicio. Es una medida del desempeño de un sistema de transmisión que refleja su calidad de transmisión y disponibilidad de servicio.

R

Red: Conjunto de computadoras, impresoras, ruteadores, switches y otros dispositivos que se puedan comunicar entre sí.
RJ-11/RJ-45 (Registered Jack): Tipos de conectores de 6 y 8 contactos usados en cableado estructuradopara telefonía (RJ-11) y redes LAN (RJ-45).
Router (ruteador): Dispositivo de la capa de red que utiliza una o más medidas para determinar la trayectoria óptima a lo largo de la cual deba direccionarse el tráfico de la red.
RTP (Real-Time Transport Protocol): Protocolo utilizado en las redes IP para transportar servicios de tiempo real como pueden ser la telefonía o videoteléfono.

S

Socket: Punto de comunicación a través del cual un proceso puede comunicarse con otro.
Señalización: Proceso que consiste en enviar una señal de transmisión a través de un medio físico para propósitos de comunicación.
Servidor: Es un nodo o programa de software que provee servicios a clientes.
SMTP (Simple Mail Transfer Protocol): Protocolo de transferencia de correo electrónico dentro del ámbito de Internet. Permite el servicio e-mail.
SNMP (Simple Network Management Protocol): Protocolo desarrollado en el ámbito de Internet para la gestión de componentes de red (routers, switches, etc).
SS7 (Signalling System 7): Sistema de Señalización número 7. Sistema CCS estándar que se utiliza en ISDN.

T

TCP (Transfer Control Protocol): Protocolo del Control de la Transmisión. Protocolo orientado a la conexión que pertenece a la capa de transporte y que ofrece una transmisión confiable de datos dúplex total.
TDM (Time Division Multiplexing): Multiplexaje por División en Tiempo. Es la técnica que permite asignar ancho de banda a la información a la información proveniente de diferentes canales en un solo cable, con base en ranuras de tiempo previamente asignadas.
TDMA (Time Division Multiple Access): Procedimiento de acceso sobre un mismo medio mediante el uso de TDM.
Tono: Señal analógica de una frecuencia específica, audible para las personas.

Transmisión Serial: Es un método de transmisión de datos en el que los bits de un carácter de datos se transmiten de manera secuencial a través de un solo canal.

U .....

UDP (User Datagram Protocol)**:** Este protocolo es no orientado a la conexión, y por lo tanto no proporciona ningún tipo de control de errores ni de flujo, aunque si que utiliza mecanismos de detección de errores. El protocolo UDP es muy sencillo y tiene utilidad para las aplicaciones que requieren pocos retardos o para ser utilizado en sistemas sencillos que no pueden implementar el protocolo TCP.
UTP (Unshielded Twisted Pair): Se trata de pares de cobre trenzados sin pantalla exterior para utilizar en cableado estructurado.

V .....

VoIP (Voice over Internet Protocol): Servicio de transmisión de señal de voz mediante el uso de paquetes en una red IP. Utiliza los protocolos RTP, SIP, H.323 y RSVP.

W .....

WAN (Wide Area Network)**:** Denominación genérica utilizada para una red de datos que ocupa un área extensa, generalmente a nivel nacional o internacional.
WWW (World Wide Web): Es una red de gran tamaño de servidores de Internet que proveen hipertexto y otros servicios a terminales que corren aplicaciones del cliente como los navegadores WWW.

BIBLIOGRAFÍA

INTRODUCCIÓN

http://computer.howstuffworks.com/ip-telephony.htm/printable

http://voipreview.org/news.details.aspx?nid=66

http://voipreview.org/how_does_work.aspx

1. REDES IP

http://eia.udg.es/~atm/tcp-ip/index.html

http://www.uv.es/~montanan/redes/redes_03.pdf

http://www.uv.es/~montanan/redes/redes_06.pdf

http://www.miliuco.net/docs/tcpip.htm

http://tau.uab.es/~gaby/DEA/1%20Los%20protocolos%20TCPIP.pdf

http://mx.geocities.com/alfonsoaraujocardenas/topologias.html

http://www.saulo.net/pub/redes/a.htm

http://www.ldc.usb.ve/~redes/Temas/Tema.14/ospf.htm

http://www.isa.uniovi.es/docencia/redes/tema6.pdf

http://teleco.spymac.com/apuntes/Varios/Biblia%20de%20Telecomunicaciones/1201.pdf

http://teleco.spymac.com/apuntes/Varios/Biblia%20de%20Telecomunicaciones/1207.pdf

http://teleco.spymac.com/apuntes/Varios/Biblia%20de%20Telecomunicaciones/1210.pdf

http://www.monografias.com/trabajos7/protoip/protoip.shtml

2. VOIP

http://www.cisco.com/global/ME/pdfs/easy_data_voice.pdf

http://w3.tmit.bme.hu/labor/meresek/voip/Voip(angol).pdf

http://www.oecd.org/dataoecd/24/5/2076710.pdf

http://www.attackprevention.com/article/Acquiring_VoIP_Knowledge_Training_Needs_and_How_to_Meet_Them-719.html

http://www.quintum.com/files/prod_lit/calling3.pdf

http://electronics.howstuffworks.com/ip-telephony.htm/printable

http://www.net.com/pdf/Integrated_Tele_Plat_wp.pdf

http://www.net.com/pdf/VoIP_Bandw_Utilz_wp.pdf

http://www.voip-news.com/rsc/3a.htm

http://documents.iss.net/whitepapers/ISS_VoIP_White_paper.pdf

http://www.smallbusinesscomputing.com/webmaster/article.php/3324261

http://www.aitel.hist.no/fag/ipt/lek02/iptel_latency_brooktrout.pdf

http://www.erlang.com/protocols.html

http://www.protocols.com/pbook/pdf/voip.pdf

http://www.imagen-interactive.com/assets/whitepapers/telcosystems_voip.pdf

http://www.ntp-systems.com/pdf/Sync_VoIP.pdf

3. PROTOCOLO SIP

Camarillo, Gonzalo. “SIP Demystified”. Ed McGraw-Hill. Segunda Ed.

http://www.rediris.es/mmedia/gt/gt2003_1/sip-gt2003.pdf

http://www.hispasec.com/unaaldia/1593

http://www.cudi.edu.mx/primavera_2005/presentaciones/rodolfo_castaneda.pdf

http://www.recursosvoip.com/protocolos/sip.php

http://www.latca.com/Notas_Tecnicas_VoIP.htm

http://www.packetizer.com/voip/sip/

http://www.intertex.se/upfiles/IntertexSIPWhitePaper.pdf

4. APLICACIÓN

VxWorks y Tornado

http://cfa-www.harvard.edu/cfa/oir/IOTA/TECHNICAL/VxWTutorial.html

http://www.graduatingengineer.com/companies/hightech/windriver.html

http://www.windriver.com/products/development_tools/ide/tornado2/tornado_2.pdf

Servidor FTP

http://www.ujaen.es/sci/redes/ftp/servftp/servftp.htm

Servidor SIP

http://www.rediris.es/rediris/boletin/65/enfoque1.pdf

López Ángel - Novo Alejandro, “Protocolos de INTERNET Diseño e Implementación en Sistemas UNIX”, Capítulo 2 Los Sockets pp 7-52. México, Alfaomega.

Manual del Teléfono “Teléfono Modular de Interiores de México” versión 03, SIEMENS.pp 3 – 18.

Ford Merilee - Lew Kim H. - Spanier Steve - Stevenson Tim, “Tecnologías de Interconectividad de Redes”, Glosario pp. 575-664. México 1998.
señalización de todos los demás canales del sistema.

5. MERCADO

Wexler, Joanie. VoIP, Awindfall of Savings, Productivity, and Applications. CISCO Systems user Magazine. 2005. Vol.25.

http://www.bloggers.com.ar/bloggers/novedades3/8119.html

http://www.conocimientosweb.net/portal/ftopic1008.html

http://www.alt1040.com/archivo/2005/04/21/se-confirma-el-bloqueo-de-voip-en-telmex-prodigy/

http://www.packetizer.com/voip/papers/understanding_voip/ngn.html

http://www.packetizer.com/voip/papers/understanding_voip/voip_protocols.html

http://www.enterate.unam.mx/

http://www.ingetel.com/Paginas/Telefonia_IP_docus.htm

http://www.voxip.com.mx/cont_p1.php

http://www.skype.com/

http://portal.avantel.com.mx/wps/portal/

http://www.voxip.com.mx/

http://www.maxcom.com.mx/