CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara de Ingeniería Avanzada

UNA ARQUITECTURA DE SOFTWARE CONSIDERANDO MECANISMOS DE SINCRONIZACIÓN PARA SISTEMAS MULTIMEDIA BASADOS EN LA RECOMENDACIÓN ITU - T H.323

TESIS QUE PRESENTA:
DAVID RAFAEL GARDUÑO BARRERA

PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE:
INGENIERÍA ELÉCTRICA

Guadalajara, Jal., Enero de 2000

Agradecimientos

A Dios. *Que me permitió llegar a cumplir mi meta*
A mis padres. *Porque con su esfuerzo y ejemplo me han impulsado a seguir adelante.*
A mi asesor Dr. Jacinto, *por ayudarme a llevar a cabo este trabajo*
A mis sinodales *Dr. José Luis Leyva y Dr. Deni Torres por su apoyo en la revisión de este trabajo*
A los profesores del CINVESTAV-GDL *y a todos los que han contribuido en mi formación académica*
A mis compañeros de carrera *que con sus charlas y bromas hacían menos pesado el trabajo*
Al CONACYT *que me apoyó económicamente durante mis estudios*

A todos ellos muchas gracias.

Becario de CONACYT número 121171

Tabla de contenido

Prefacio

Los medios de comunicación actuales tienden a ser más naturales cada día, esto es, tienden a parecerse más a una interacción directa con otra persona, en esta interacción podemos percibir los gestos, sonidos, olores y texturas de nuestro alrededor y nuestro interlocutor.

El ideal al establecer una comunicación a través de un medio, es que lo hagamos con la misma naturalidad que cuando nos encontramos frente a frente con nuestro interlocutor, pero además, aprovechando las propiedades de tratamiento de la información que ofrece el uso de las redes de computadoras. Un sistema con estas características no solo podría mostrar voz e imagen, sino información relacionada que permita un refuerzo en la comunicación.

A estas expresiones y la forma en que se transmiten les llamamos medios de comunicación, por ejemplo: a una carta, un periódico, una revista o un papel volante se les llama medios de comunicación escritos. A la radio y el teléfono se les llama medios de comunicación orales o auditivas. A un sistema que combina los medios anteriores más imágenes, presentaciones, y potencialmente olores o emociones, se le llama Sistema Multimedia.

Nos encontramos ahora en la búsqueda de formas de comunicación que nos permitan integrar varios medios de expresión en uno solo para transferir información entre lugares remotos y deseamos que estas formas sean lo más rápidas y naturales posibles.

A la comunión de varios medios de comunicación se les conoce como sistemas multimedios, o con la palabra **MULTIMEDIA**, de raíces latinas:

MULTI: varios

MEDIA: formas, tipos, medios

Estos sistemas multimedia combinan una variedad de fuentes de información tales como: voz, gráficos, animación, imágenes, audio y video en un amplio rango de aplicaciones [10].

Tales sistemas pueden ser utilizados para transmitir información entre dos entes geográficamente próximos, pero si deseáramos que esta información fuera presentada a un ente distante, tendríamos que hacer uso de otras herramientas para transferir tal información. En nuestro caso, utilizaremos las **Redes de Computadoras.**

Una red de cómputo es un conjunto de computadoras autónomas interconectadas entre sí. Por autónomas decimos que pueden ejecutar su proceso de manera independiente pero preservando la capacidad de cooperar para alcanzar un fin común. Por interconectadas, queremos decir que tienen la capacidad de intercambiar información entre ellas [16].

Debido a su facilidad de uso y la reducción en su costo, las computadoras son una buena herramienta para la captación, representación, procesamiento, almacenamiento y presentación de información con formato multimedia; pero para lograr nuestro objetivo, transmitir información entre entes distantes, es necesario hacer uso también **de redes de computadoras**, lo que incrementa la complejidad de este trabajo. La naturaleza de la transmisión actual de información por medio de redes de cómputo generalmente impone un compromiso entre la velocidad y la calidad de la información.

Como definición general, podemos decir que un sistema tiene características de sincronización si asegura un orden temporal de eventos [8].

Las propiedades de los sistemas multimedia implican que un medio sea reproducido manteniendo las características temporales dadas en su generación, esto es, que no existan retrasos ni traslapes en la presentación de los datos en un solo medio mientras que, si existen dependencias o relaciones temporales entre dos o más medios, estas sean también respetadas.
Si estos sistemas de comunicación no tuvieran características de sincronización, los datos presentados serían no aptos, confusos y desagradables para el receptor, evitando así que el emisor transmita su mensaje en su totalidad.

Organización.

El trabajo aquí presentado está organizado de la siguiente manera:

- Capítulo 1. Introduce al resto del trabajo y presenta algunos términos necesarios para el mejor entendimiento de este documento.
- Capítulo 2. Define el concepto de multimedia y las características de los flujos multimediales, así como los tipos y propiedades de estos flujos, muestra además una definición para la sincronización y algunas de sus propiedades.
- Capítulo 3. Da una breve definición del problema a resolver: modificar H.323

- Capítulo 4. Muestra formalmente la especificación de requerimientos de software para el problema actual.
- Capítulo 5. Da una descripción de los sistemas telefónicos actuales y los servicios que ofrecen. Muestra también una descripción de la recomendación ITU-T H.323 y el lenguaje de especificación LOTOS.
- Capítulo 6. Describe el lenguaje LOTOS y una metodología propia desarrollada para el análisis y desarrollo de protocolos (LAPEM)
- Capítulo 7. Muestra el algoritmo de sincronización seleccionado y la arquitectura propuesta
- Capítulo 8. Formaliza y modela la arquitectura propuesta para la implantación de la recomendación ITU-T H.323 modificada para el problema específico.
- Capítulo 9. Concluye el trabajo, describe las características de la arquitectura propuesta, da una descripción de las contribuciones o innovaciones logradas y enumera los trabajos futuros propuestos.
- Apéndice A. Muestra el ejemplo completo referenciado en el capítulo 7.
- Apéndice B. Muestra el listado de código LOTOS que describe el comportamiento de la arquitectura propuesta.

1

Introducción

Antecedentes

Desde su invención, el teléfono sufre cambios radicales en ciclos de 10 años. En los 50's, la introducción de cables coaxiales trasatlánticos permitió llamadas internacionales de manera directa. En los 60's, la transmisión digital y la conmutación incrementaron drásticamente la calidad del audio; en los 70's, los conmutadores programables permitieron la digitación por tonos y algunos servicios digitales como la llamada en espera; y en los 80's la expansión e implementación de sistemas de señalización de canal común fuera de banda, hicieron algunos servicios como los números 800 (servicio de telefonía gratuita) posibles.
Estos cambios definen una trayectoria desde la señalización transmisión análoga, hacia la digital, transmisión de circuitos conmutados y señalización basada en paquetes.

En los 90's, la telefonía vía Internet, marca el último paso de este largo camino a una infraestructura de servicios de comunicación digital integrada [17]. Más aún, se estima que el siguiente paso sea la comunicación multipunto de estos servicios y la adición de otras características, como video y datos, entre otros.

El número creciente de aplicaciones de las comunicaciones multimedia aumenta a pasos agigantados, ya no son solamente textos e imágenes, se han agregado a estos el video, audio y realidad virtual (una combinación de los anteriores), así como datos discretos como textos y gráficos.

Multimedia, desde el punto de vista de un usuario final de computadoras, significa que la información que viene del ordenador puede ser presentada de varias maneras como: imágenes, textos, sonidos, videos o una combinación de los anteriores, obteniendo así una mejor comprensión de la información. Por ejemplo, una lección de química puede ser expresada mejor si vemos átomos o moléculas de los elementos unirse y escuchamos una onomatopeya de reacciones químicas mientras una voz nos explica el tipo de reacción, elementos y resultados del experimento. La misma lección podría ser mostrada al usuario mediante solo un texto e imágenes ilustrativas, teniendo con esto un aprendizaje más lento y confuso.

Adicionalmente, esta presentación multimedia puede ser transmitida a sitios remotos, mediante redes de telecomunicaciones y de cómputo, permitiendo así que la información pueda llegar a más personas. Esto implica la aplicación de otras áreas de investigación, como son: sistemas distribuidos y trabajo cooperativo (CSCW)

A esta información multimedia transmitida por redes se le puede agregar la capacidad de permitir que los usuarios interactúen con el sistema o con otros usuarios con herramientas de este tipo. Esto nos lleva a incluir a los sistemas de tiempo real y técnicas de coordinación y sincronización en este trabajo.

Al momento del desarrollo de este trabajo, existen una gran cantidad de empresas, instituciones y universidades tratando de resolver problemas similares, así como aplicaciones para herramientas como estas.

1.1 Areas de aplicación.

Se puede utilizar la combinación de multimedia y telecomunicaciones en áreas de aplicación muy variadas y con solo los límites que la imaginación nos trace. En la actualidad se pueden encontrar varias de estas aplicaciones, entre ellas están:

- Educación: la mayoría de las lecciones que se imparten en las escuelas podrían ser presentadas más fácilmente por los profesores y entendidas con mayor facilidad por los estudiantes en un formato multimedia, además que podrías ser aprovechados por una cantidad mayor de alumnos con el uso de redes de comunicaciones.

- Trabajo: se desarrollan actualmente proyectos de trabajo cooperativo [14] que hacen uso de multimedia y redes de cómputo para permitir la colaboración de varios usuarios en proyectos comunes.

- Investigación: muchas son las áreas de la ciencia que requiere de simulaciones. Anteriormente los resultados de tales simulaciones eran difíciles de interpretar, pero con la aparición de multimedia, los resultados son gráficos y en ocasiones auditivos, lo que mejora su interpretación. Otra aplicación que se encuentra en estado de experimentación son los laboratorios virtuales [47].

Pero este gran desarrollo no hubiera sido posible si no existieran intereses de por medio. Debido a esto, existen varias ramas de la industria que con su desarrollo han impulsado nuestro tema de interés. Entre estas industrias se encuentran:

- Estudios de grabación y productoras de TV: estas empresas son pioneras en la edición digital de audio y video. Algunos de estos sistemas son extensiones de formatos de computadoras, pero acelerados por tarjetas especiales. Sus usuarios son cada vez más exigentes, y por esto requieren de mejores programas y equipos.

- Telecomunicaciones: iniciaron principalmente con la aparición del teléfono. Las redes telefónicas migraron gradualmente a redes digitales, que son similares a las redes de computadoras. Actualmente, las computadoras hacen el trabajo de la conmutación de llamadas en las centrales telefónicas. De esta manera, los teléfonos son, cada día más, parte de las computadoras.

- Electrónica: la aparición constante de nuevas redes más rápidas, switches más veloces, dispositivos de producción y reproducción con mayor calidad y velocidad, medios de almacenamiento con mayor capacidad y rapidez, además de la baja sustancial en sus costos, ha provocado que los sistemas de comunicación puedan llegar a una cantidad mayor de personas, incrementando así su demanda [13].

1.2 Sistemas en tiempo real.

Desde del punto de vista técnico, además del manejo de una gran cantidad de información, los requerimientos de tiempo sobre todos los datos es uno de los mayores problemas a resolver. Las computadoras tienden a resolver su trabajo tan pronto como sea posible es decir, haciendo su mejor esfuerzo. Esto, en comunicaciones multimedia, no es suficiente, se debe asegurar un tiempo máximo de espera para que los datos sean entregados y reproducidos. Para esto deben desarrollarse estrategias o algoritmos que permitan que los datos viajen a través de las redes de una manera rápida y confiable.

El Instituto Nacional Alemán para la Estandarización (DIN por sus siglas en alemán, similar al ANSI en América) define un proceso en tiempo real en una computadora como sigue:

> *Un proceso en tiempo real, es aquel que entrega el resultado de su procesamiento en un lapso de tiempo dado.*

La característica principal de los sistemas de tiempo real es la corrección de la computación. Esta corrección no aplica solamente a resultados libres de errores, sino también al tiempo en que estos resultados son presentados.
Debido a esto, los sistemas en tiempo real pueden fallar no solamente si una falla masiva de software o hardware ocurre, sino también si el sistema es incapaz de ejecutar un trabajo crítico a tiempo. El comportamiento determinístico del sistema se refiere a la adherencia del sistema a espacios de tiempo definidos.

La velocidad y la eficiencia no son, como mucha gente mal interpreta, las características principales de los sistemas de tiempo real. En una planta petroquímica, por ejemplo, el resultado es inaceptable no únicamente cuando la maquinaria de un evento responde muy rápidamente, sino cuando lo hace muy retrasada. Otro ejemplo de esto es la presentación de un video, si este se presenta muy rápido, el resultado es tan indeseable como si se presentara de forma muy lenta. Esto se ilustra en la figura 1.1

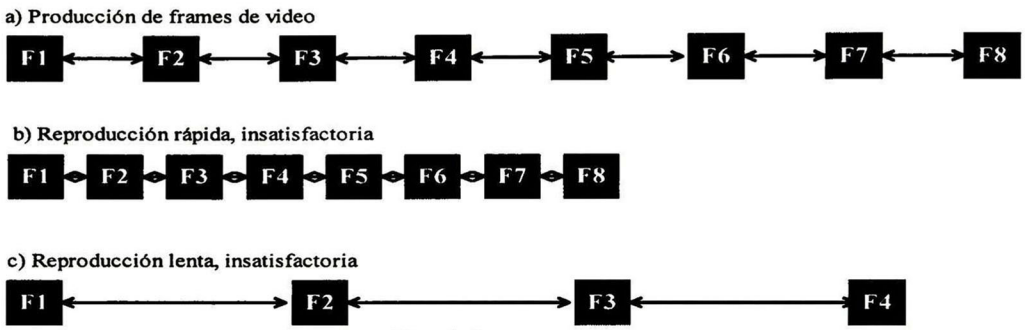


Fig. 1.1
Presentación de frames de video

Un sistema de tiempo real se distingue por las siguientes características:

- Tiempo de respuesta predecible a eventos críticos en tiempo e información oportuna de dicho evento. Por ejemplo, en el sistema de control de una planta nuclear, la respuesta a un mal funcionamiento debe ocurrir dentro de periodos de tiempo bien definidos para prevenir desastres potenciales.
- Un alto nivel de ordenamiento. El ordenamiento se refiere al grado de utilización de los recursos en los que, o bajo los que, el límite de tiempo de cada evento crítico puede ser tomado en cuenta. Esto en inglés se conoce como *Schedulability* [6], [7], [11], [20].
- Estabilidad bajo sobrecargas pasajeras. Ante una sobrecarga del sistema, el procesamiento de los eventos críticos debe ser asegurado. Estos eventos críticos son vitales en la funcionalidad básica que provee el sistema.

Los flujos de audio y video consisten de valores individuales de datos continuos, que cambian periódicamente. Cada unidad lógica debe ser presentada en un espacio de tiempo bien definido. Una pieza musical, por ejemplo, debe ser reproducida a una velocidad constante [13].

1.3 Sincronización.

Otro problema a resolver es la integración requerida por los diferentes medios en aplicaciones multimedia. En tales aplicaciones, los medios que no dependen del tiempo, tales como textos e imágenes, así como los medios que si dependen del tiempo, audio y video por ejemplo, deben ser procesados de manera coherente. Estos medios no son independientes unos de otros, debido a esto, la integración de medios requiere de

conceptos, que son más complejos que la simple integración, Se deben definir términos como sincronización Intraflujo y sincronización Interflujo, estos conceptos se presentan en este trabajo en capítulos posteriores. Estos son útiles para entender las relaciones temporales que existen entre elementos de un mismo medio y entre elementos de varios medios.
La palabra sincronización se refiere al tiempo. La *sincronización* en los sistemas multimedia se refiere a las relaciones temporales existentes entre objetos multimedia en los sistemas multimedia. Es posible diferenciar entre objetos dependientes y no dependientes del tiempo. Un objeto dependiente del tiempo es presentado mediante un flujo.

Existen relaciones temporales entre unidades consecutivas del medio[13]. Si la duración de todas las unidades de tiempo de un flujo son iguales, llamamos a este flujo *continuo.* Ejemplos de esto son el audio y el video. En caso de que las diferencias temporales entre las unidades de presentación de un flujo no sean iguales, llamaremos a este flujo *asíncrono* o *discontinuo* Fig. 1.2.

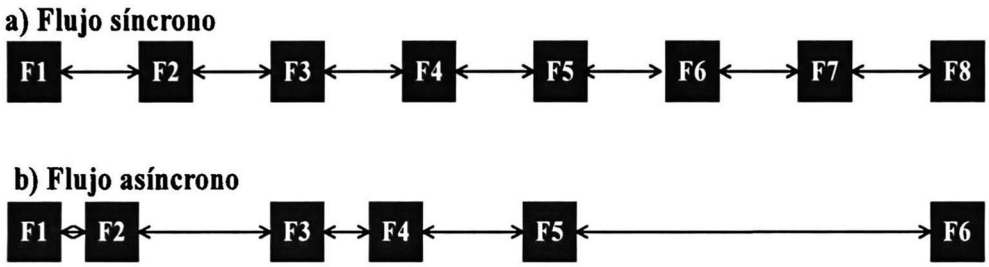


Fig. 1.2
Flujo sínclono y Flujo asíncrono

Otras descripciones de sincronización, así como de algunas metodologías para lograrla pueden ser encontradas en [11], [8], [20], [6], [10]

1.4 Objetivo

El objetivo de este trabajo es proponer una arquitectura de software basada en la recomendación ITU-T H.323 (Multimedia sobre redes IP) que sea capaz de soportar mecanismos de sincronización multimedia en redes no confiables.

Para el cumplimiento de este objetivo es necesario completar las siguientes submetas:
- Definir el problema
- Formalizar los requerimientos del problema
- Estudiar y seleccionar las herramientas de trabajo necesarias. (*LOTOS y H.323*)
- Estudiar y escoger un algoritmo de sincronización adecuado.
- Escoger los servicios digitales a ser incluidos.
- Analizar y escoger herramientas para el diseño de protocolos de comunicaciones.
- Proponer una modificación a los modelos establecidos de la especificación ITU-T H.323 para la agregación de los algoritmos de sincronización.

- Proponer, formalizar y modelar una arquitectura de software que cumpla con los requerimientos establecidos.

1.4.1 Justificación:

La búsqueda de alternativas de comunicación telefónica es de gran relevancia social y económica. Dentro de las soluciones más factibles se encuentra la transmisión de tráfico multimedia sobre redes IP.
Esta alternativa se rige por la recomendación ITU-T H.323, la cual deja abierta la definición de una arquitectura de sistema (HW y SW). De igual manera, el problema de la sincronización multimedia (asegurar la coherencia temporal de la información multimedia) no es tratado en esta recomendación.
En este contexto consideramos necesario definir una arquitectura de software adecuada para soportar todos los aspectos de tráfico multimedia en redes de tipo IP (no confiables) con especial atención al problema de la sincronización.
Como resultado de esta investigación, se propone una arquitectura de software capaz de cumplir con estos objetivos. Esta arquitectura se obtuvo mediante la aplicación de una metodología de diseño de protocolos basada en LOTOS. Esta metodología es progresiva y permite el descubrimiento de los elementos de la arquitectura y verificarla mediante el lenguaje LOTOS.

1.5 Lecturas relacionadas.

Algunos de los temas tratados en este trabajo pueden ser encontradas en algunos textos tradicionales como revistas, libros y congresos, y tal vez con mayor detalle, aunque se encuentran en temas separados. Este trabajo contiene, además y como parte principal, ideas, desarrollo y resultados propios.

En la rama de multimedia, los principales textos son editados por la IEEE, como son: "IEEE Multimedia Conference" (la primera en Mayo de 1994, Boston, MA) y "IEEE Multimedia Magazine" (desde principios de 1994), aunque también se pueden encontrar: "ACM Multimedia Conference" (la primera en Agosto de 1993, Anaheim, CA), ACM Springer journal, "Multimedia Systems" (a finales de 1993), "Multimedia Tools and Applications" (principios de 1995).
En el área de las redes de computadoras, se puede consultar con revistas publicadas por la IEEE, como son: "IEEE Internet computing", "IEEE networking", además de varios libros de texto como: Redes de computadoras de Aaron Tanenbaum, Tecnologías Emergentes de computadoras, TCP/IP de Commer.

2

Conceptos

Multimedia.

El término multimedia fue utilizado por primera vez en los años 60's para describir las presentaciones que consistían de imágenes fotográficas y audio casetes, aunque el arranque verdadero de los sistemas multimedia fue a finales de los años 70's, con el lanzamiento del *Philips Laservision videodisc*. Aunque estos sistemas eran de tipo analógico, ya combinaban audio, imágenes estáticas e imágenes en movimiento.

La aparición del *Laservision* permitió la creación de la primer aplicación de video interactiva. El *Laservision* también permitió el nacimiento del disco compacto de audio digital en 1982. Así nacieron el CD-ROM y el CD-I (*compact disc interactive*). Podemos aquí notar el cambio gradual de los sistemas analógicos a los sistemas digitales [15].

2.1 Medio

En general, podemos describir un medio como una forma de distribución y presentación de información [13]. Ejemplos de esto, son: textos gráficos, música, voz, cables, papeles, transportes, etc.
Los medios pueden ser clasificados de diversas maneras y con respecto a diferentes criterios. Aquí presentamos algunos criterios de clasificación de medios.

2.1.1 Percepción

La pregunta central de esta clasificación es: *¿Cómo un ser humano percibe la información?* El medio de Percepción ayuda a los humanos a sentir su medio ambiente. Este medio se refiere a la naturaleza de la información tomada como el ser humano la recibe. De esta manera, los medios pueden ser clasificados como: visuales, auditivos, táctiles, etc.
Es importante saber como un ser humano recibe información desde una computadora. La mayoría de la información se transmite a través de imágenes y sonidos, pero en la actualidad, se están desarrollando formas de transmitir información utilizando otros sentidos, por ejemplo el tacto.

Estos sistemas se desarrollaron con la finalidad de transmitir información, y esta información puede tener muy diversas formas de expresión. Consideremos las siguientes formas:

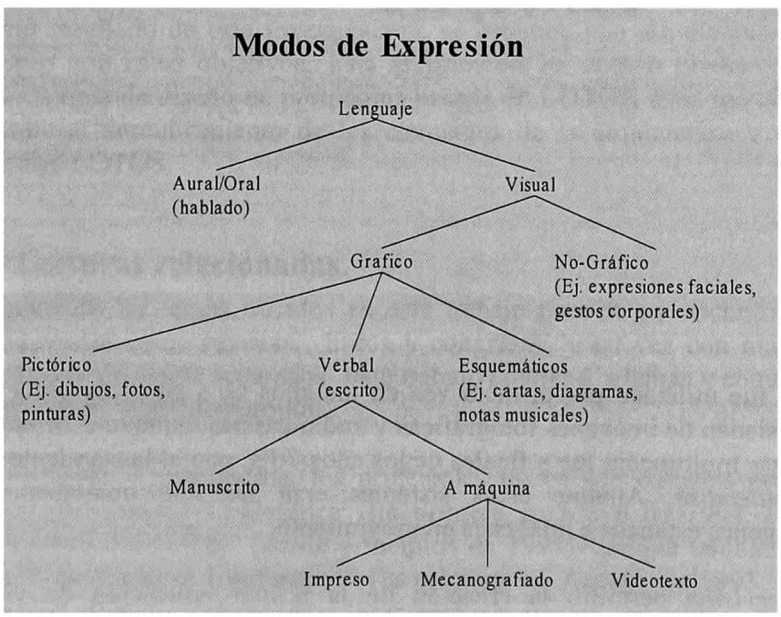


Fig. 2.1

A cada una de estas formas de expresión, le llamamos, medio.

2.1.2 Representación.

Esta clasificación responde a la pregunta: *¿Cómo se codifica la información en una computadora?*
La respuesta está dada por los diferentes formatos de almacenamiento y codificación de datos en una computadora, por ejemplo:

- Los textos pueden ser almacenados en formato ASCII o EBDIC, entre otros.
- Los gráficos pueden ser codificados como GIF, JPEG, entre otros.

- El audio puede tener formatos como PCM, AU, MP3, WAV, etc.

2.1.3 Presentación

La pregunta correspondiente a esta clasificación es: *¿Por qué medios físicos se reproduce la información?*
Se puede utilizar papel, bocinas, monitores, teclado y ratón (en una computadora), etc.

2.1.4 Almacenamiento

Se refiere al elemento físico en el que la información es guardada, pero no está limitada a los elementos presentes en una computadora. De esta manera, el papel, una cinta magnética, un microfilm, un disco duro, un floppy disk, entre otros, son medios de almacenamiento.

2.1.5 Transmisión

Los medios de transmisión caracterizan a diferentes elementos físicos que permiten una transmisión continua de datos, de manera más exacta, que permiten la transmisión de señales de telecomunicaciones. Elementos de estos medios son: las redes cableadas, fibra óptica, y el aire (en caso de microondas).
Nótese que se excluyen los medios de almacenamiento debido a la definición de medios de transmisión.

2.1.6 Tiempo

Cada medio puede encontrarse en una o más dimensiones de representación. Por ejemplo, un gráfico en pantalla tiene dos dimensiones espaciales, pero la holografía y estereofonía requieren de una dimensión más.

El tiempo es una de las dimensiones que pueden afectar a los medios. Con respecto al tiempo, los medios pueden ser de dos tipos:

- Los medios que consisten de una secuencia de elementos sin un componente temporal, se dice que son *discretos o no dependientes del tiempo*. Ejemplos de este tipo de medios son los textos y los gráficos. Este tipo de medios debe ser procesado tan rápido como sea posible, pero sin que su retraso signifique un error crítico, ya que la validez de los datos no depende de alguna condición temporal.
- En otro extremos tenemos a los datos que, además de tener una secuencia de elementos, tienen una componente temporal. Este tipo de medios cambian con en tiempo y su información es expresada, no solamente por los valores individuales de sus elementos, sino por su ocurrencia en una línea de tiempo. Este tipo de medios se conocen como *dependientes del tiempo* (continuos). Ejemplos de medios de este tipo son el audio y el video.

Los *no dependientes del tiempo* no tienen periodicidad en el espacio de tiempo entre sus elementos, mientras que los *dependientes del tiempo* si la tienen.

2.2 Propiedades de los Sistemas Multimedia

Si tomáramos una definición etimológica de multimedia, podríamos decir que un sistema multimedia es aquel que soporta al menos un tipo de medio. Aunque esto no es una definición adecuada al uso del término que tendremos en este trabajo, ya que solamente es una definición cuantitativa del sistema [13]. Tomando esta definición, un procesador de textos que mezcla textos y gráficos es un sistema multimedia, pero este tipo de sistemas existen antes de que el término fuera concebido en los sistemas de cómputo. Debido a esto, es necesario una nueva definición del término para el caso de los ambientes computacionales.

Una nueva concepción, podría incluir características cualitativas. De esta manera, la definición incluiría el tipo de medio que se está manejando.
Los sistemas multimedia se distinguen de otros por varias características. Se mencionan a continuación algunas de las propiedades más importantes que caracterizan a este tipo de sistemas.

2.2.1 Combinación de medios

No cualquier combinación arbitraria de medios justifica el uso del término multimedia. Con la definición básica de combinación de medios, un procesador gráfico o un procesador de textos podría ser llamado un sistema multimedia. En este trabajo se tomarán solo los casos en que existan al menos un medio continuo o dependiente del tiempo.

2.2.2 Independencia

Comúnmente los medios tienen cierto nivel de independencia entre ellos. El caso de interés de este trabajo es cuando existen relaciones semánticas entre los flujos, desde el punto de vista del receptor de la información. Estas relaciones semánticas pueden ser cumplidas mediante el cumplimiento de relaciones temporales entre elementos de los flujos. Estas características se formalizan mas adelante.

2.2.3 Soporte de computadoras

Los medios son inherentemente independientes, pero las computadoras son la herramienta ideal para crear relaciones entre estos medios. Un usuario, el que desea transmitir la información, decide que creando relaciones entre varios medios, la información transmitida será fácilmente interpretable. Estas son las relaciones semánticas que se mencionan en el punto anterior.

2.2.4 Comunicaciones

Pensar en un sistema multimedia en el que se limitara la captura, generación, procesamiento y reproducción a una sola computadora aislada, sería poco creativo y un

desperdicio de tecnología. En la actualidad, la tendencia común es interconectar todas las computadoras, logrando de esta manera una compartición de información benéfica para todos. Por esto, la capacidad de comunicación es deseable en los sistemas multimedia.

2.3 Multimedia

2.3.1 Definición

La definición de multimedia dada hasta el momento es insuficiente.
Los sistemas Multimedia combinan una variedad de fuentes de información, tales como voz, gráficas, animación, imágenes, audio y video dentro de una amplia rama de aplicaciones [10].

La definición de multimedia que tomaremos en este trabajo será dada a partir de las características de los sistemas multimedia antes mencionadas.

Definición: Multimedia.
Multimedia es la conjunción de dos o más medios, con al menos uno de ellos dependiente del tiempo (continuo), existen relaciones temporales estrictas entre al menos dos de los medios y cuentan con un sistema de comunicación que permite transmitir los datos multimedia a otra entidad.

Otras definiciones de multimedia pueden ser encontradas en [11], [8], [20], [6], [10], [13].

2.3.2 Clasificación

Existen varios criterios de clasificación de sistemas multimedia de los que pueden ser distinguidos tres: el número de medio, el tipo de medios y el grado de integración[13].
El criterio más simple es el número de medios utilizados en una aplicación. Con este criterio, un procesador de textos que soporte texto y gráficos, es un sistema multimedia. La siguiente figura muestra un sistema multimedia que cumple con la clasificación anterior.

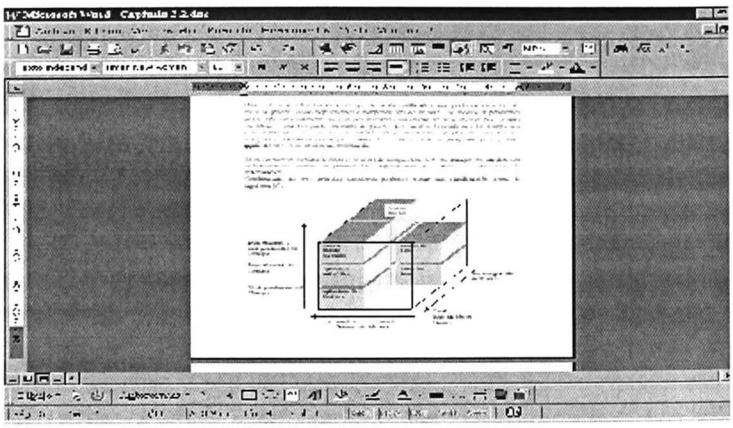


Fig. 2.2
Solo dos medios

Otro criterio de clasificación es el tipo de medios utilizados, que podemos clasificar de manera genérica como dependientes e independientes del tiempo. Los medios dependientes del tiempo son usualmente presentados utilizando una unidad de presentación, por ejemplo un dibujo o una fotografía en mapa de pixeles. Los medios dependientes del tiempo son presentados por una secuencia de unidades de presentación. Ejemplos de esto son una imagen en movimiento o una pieza musical. De esta manera, un programa que reproduce audio de un CD es un sistema multimedia.

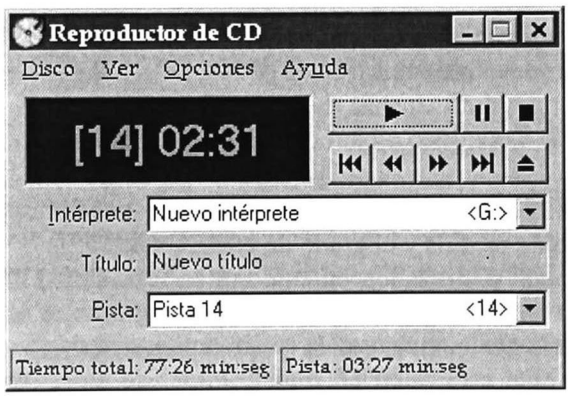


Fig. 2.3
Solo dos medios

El tercer criterio de clasificación es el nivel de integración. Esto es, aunque los medios son independientes, pueden ser presentados conjuntamente para facilitar la transmisión de información.
Combinando los tres criterios anteriores podemos tomar una clasificación como la figura 2.4 [6]:

De acuerdo con esta clasificación, el trabajo de esta tesis se ubicó en las siguientes secciones: Sistemas Digitales Integrados, Aplicaciones Audio Video y Servicios de Video.

Los sistemas digitales integrados pueden soportar todos los tipos de medios, y debido al procesamiento digital, pueden proveer un alto nivel de integración de medios.

Los sistemas que manejan objetos dependientes del tiempo en forma análoga y objetos no dependientes del tiempo en forma digital, son llamados *híbridos*. La desventaja de los sistemas híbridos es que están restringidos en integración de medios, ya que los medios dependientes del tiempo son almacenados en dispositivos diferentes que los medios independientes del tiempo (Fig. 2.5).

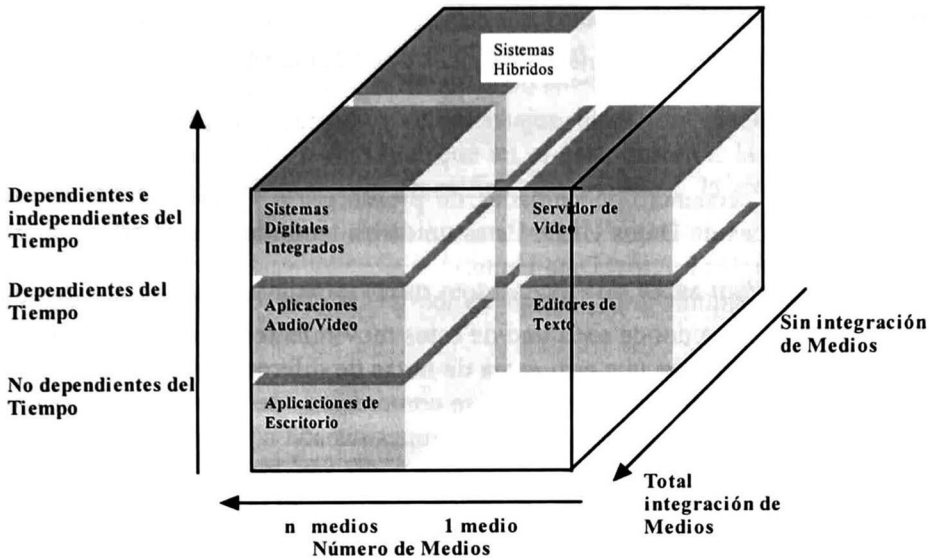


Fig. 2.4
Clasificación de sistemas multimedia

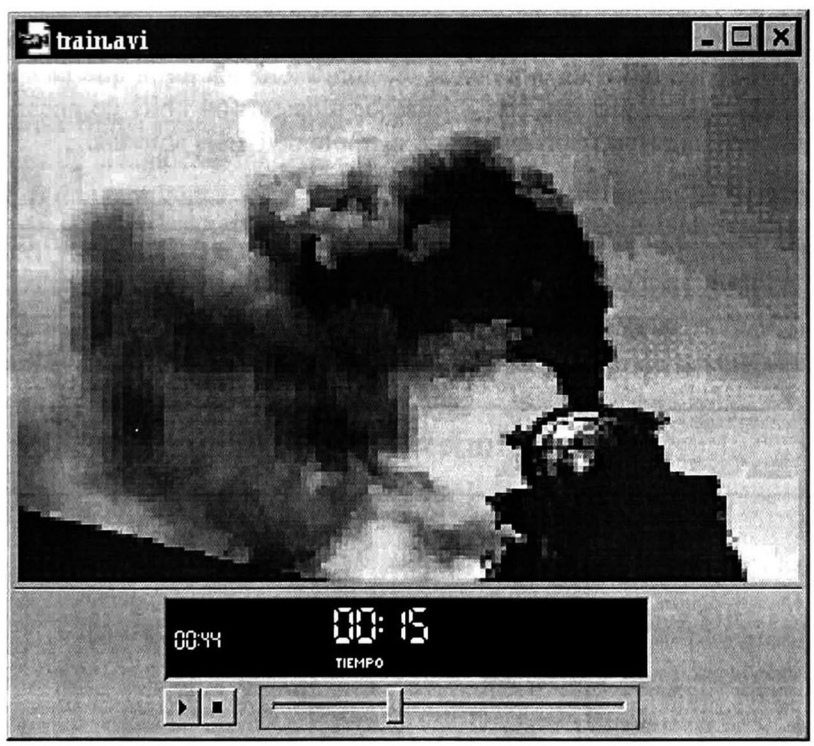


Fig. 2.5
Sistema multimedia completo

2.4 Características de los medios.

Un medio es una conjunto ordenado de unidades lógicas de presentación

2.4.1 ULDs

Un medio consiste de una secuencia de unidades de presentación. A estas unidades les llamaremos Unidades Lógicas de Datos ULD. Estas unidades también se les conoce como LDU [6] por sus siglas en inglés Logical Data Units.
Normalmente se permite cualquier granularidad en los ULDs. Por ejemplo, una sinfonía está compuesta por movimientos donde cada uno de estos movimientos es una composición independiente. Cada uno de estos es una secuencia de notas de diferentes instrumentos. En el caso de calidad de CD con codificación PCM y sin compresión, se tiene un muestreo de 44100 Hz con dos canales de 16 bits. En un CD, estas muestras son combinadas en bloques de 1/75 s [13]. Esto se puede ver en la figura 2.6.

La granularidad aquí puede variar. La sinfonía completa puede ser tomada como un ULD, pero también podemos decir que un ULD es cada uno de los movimientos de la sinfonía, o en caso deseado, se puede tomar cada una de las muestras de audio como un ULD.
El nivel de granularidad de los ULDs puede establecerse de acuerdo a nuestras necesidades. Adicionalmente, los ULDs pueden ser clasificados como abiertos o cerrados[6]. Los ULDs cerrados tienen una frecuencia y duración predecible mientras que los ULDs abiertos no. Un ejemplo de ULDs cerrados es una comunicación telefónica en la que la frecuencia de muestreo es de 125 microsegundos con 8 bits por muestra (64 kbits de ancho de banda) mientras que una comunicación tipo *chat* es un ejemplo de ULDs abiertos.

Diferentes niveles de granularidad para ULD's

Fig. 2.6

2.4.2 Terminal.

Una terminal es cualquier entidad A comunicante con las siguientes características.

i) Potencialidad de generación de ULDs multimedia.
ii) Capacidad de reproducción de ULDs multimedia.
iii) Capacidad de transmitir mensajes discretos hacia alguna otra terminal B, pudiendo ser B = A. La forma de interconexión de ambas terminales no se especifica, asumiéndose la existencia de un canal entre ambas.
iv) Capacidad de recepción de mensajes discretos desde alguna otra terminal C, pudiendo ser C=A. Al igual que en el caso anterior, la forma de interconexión de ambas terminales no se especifica, asumiéndose la existencia de un canal entre ambas.

En este trabajo, las terminales serán representan con letras mayúsculas, iniciando desde *A*, *B*, *C*,... ,*Z*

2.4.3 Flujo

Llamaremos flujo a una secuencia unidireccional de ULDs ordenada temporalmente. Sea ***x*** un flujo, entonces ***x*** se define de la siguiente manera: $x = \{ULD_0, ULD_1, ..., ULD_n\}$

Tal que $n \in N_0$. Los flujos se representan usando letras minúsculas comenzando con las últimas letras del alfabeto, de esta forma *z, y, x,. .., a.*
Un flujo siempre está relacionado con dos terminales A, B cualesquiera, uno de los cuales llamaremos ***el origen***, y a la otra ***el destino*** del flujo.

2.4.4 Sesión

Son uno o más flujos multimedia todos en la misma dirección entre dos terminales. Los representamos de la siguiente manera:

Sean *A*, *B* dos terminales, sean $x_1, x_2, ..., x_n$ flujos con dirección de A hacia B y sea *c* un conjunto de *m* flujos $c_1, c_2, ..., c_m$ de *B* hacia *A* posiblemente vacío(*m* = 0). Entonces la sesión entre *A* y *B* se representa $A \xrightarrow[x_1, x_2, ..., x_n, c]{} B$. El conjunto *c* representa los flujos de datos de control que pueden o no existir en una sesión dada. Para simplicidad de notación, el conjunto ***c*** es obviado y por lo tanto suprimido de la notación.

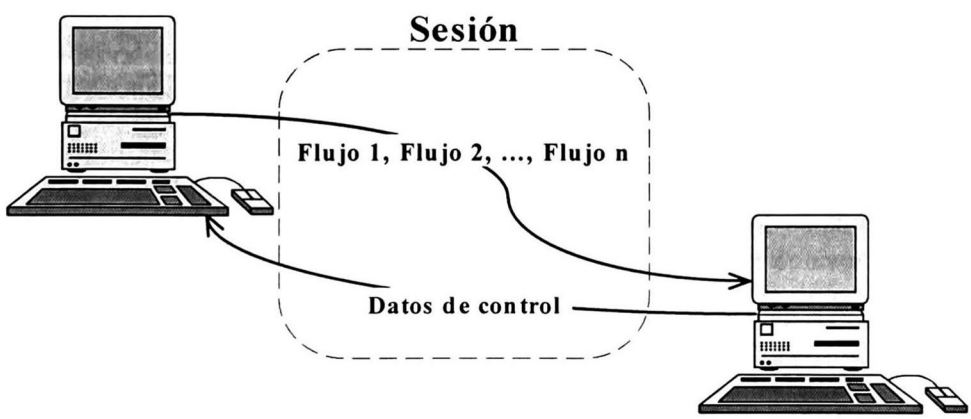


Fig. 2.7

Entonces, una sesión es uno o más flujos con la misma dirección y entre las mismas terminales. Esto se muestra en la figura 2.7.

2.4.5 Flujos Síncronos

Este flujo define un retraso máximo de terminal a terminal para cada paquete del flujo de datos. Este límite superior no debe ser nunca violado. Más aún, un paquete puede alcanzar a la terminal destino en cualquier tiempo menor que el establecido. Es decir, se puede garantizar un retraso máximo de terminal a terminal.
Adicionalmente, una conexión de audio puede ser establecida sobre una red de área local mediante un flujo síncrono. Esto lo vemos en la figura 2.8.

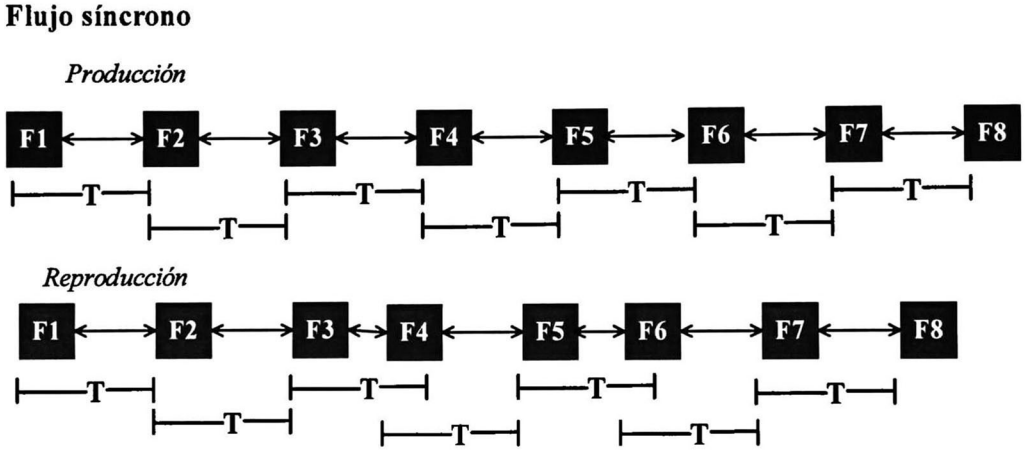


Fig. 2.8

2.4.6 Flujos Asíncronos

Un flujo asíncrono es aquel que no tiene restricciones temporales, los ULDs son procesados tan pronto como sea posible (mejor esfuerzo). Los protocolos de Internet para correo electrónico son un ejemplo de esto. En el ámbito de las redes de área local, Ethernet es otro ejemplo. Toda la información de medios discretos puede ser transmitida mediante un flujo asíncrono. Los datos de los medios discretos pueden también incluir restricciones de tiempo a través de conexiones temporales para sincronización con medios continuos (ver fig. 2.9).

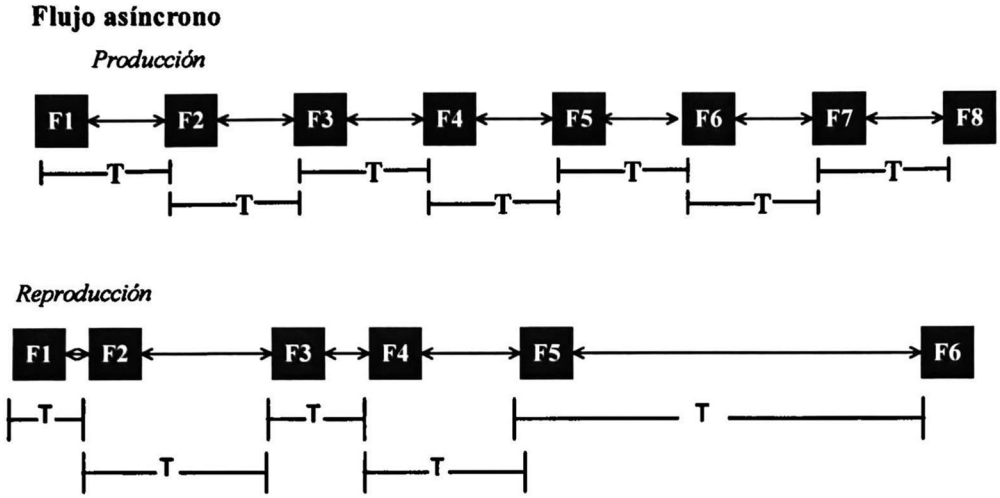


Fig. 2.9

2.4.7 Flujos Isócronos

Los flujos isócronos son un subconjunto de los flujos síncronos.

Un flujo isócrono es aquel cuyos ULDs guardan relaciones temporales y secuenciales estrictas tales que:

> Dados dos ULDs P_i, P_{i+1} que pertenecen a un mismo flujo y donde $i \in N_0$ marca la *secuencia de producción* de los ULDs, definamos $\lambda = t(P_{i+1}) - t(P_i)$ y k una constante arbitraria que representa el lapso mínimo de generación de 2 ULDs pertenecientes al mismo flujo.
>
> El flujo isócrono se caracteriza por
> $\lambda \to k$
> (lambda "tiende" a k).

El término isócrono es aplicado a un flujo en el cual los datos son generados de manera continua y periódica.

En el caso idóneo, $\lambda = k$, lo que nos indica que entre cualesquiera dos ULDs contiguos de un mismo flujo isócrono existe un tiempo constante y exacto.

A diferencia de los flujos síncronos que definen solamente un límite superior en el retraso terminal-terminal, los flujos isócronos definen también un límite inferior de retraso terminal-terminal [13] (ver figura 2.10).

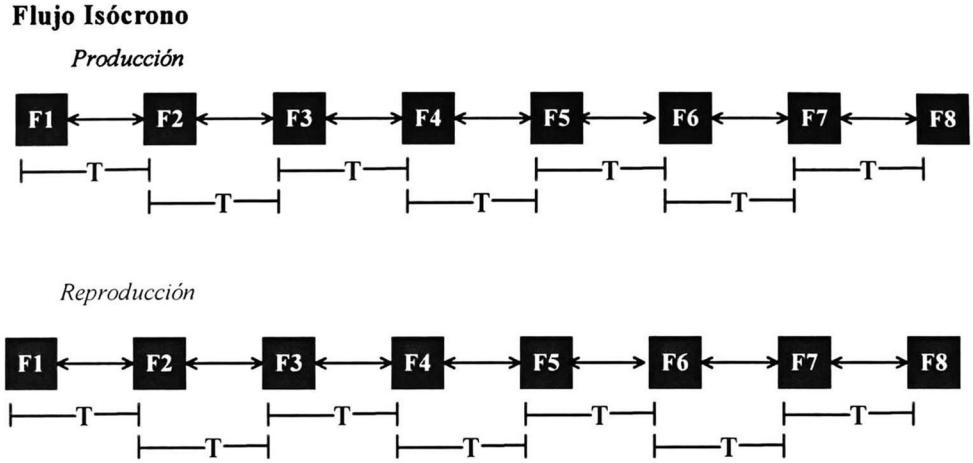


Fig. 2.10

2.5 Sincronización

Los sistemas multimedia avanzados son caracterizados por la integración, generación, almacenamiento, comunicación, manipulación y presentación de medios dependientes e independientes del tiempo, todo esto controlado por computadora [6]. El punto principal que provee esta integración es la representación digital de cualquier tipo de medio y la *sincronización* de y entre varios tipos de medios.

La palabra sincronización se refiere al tiempo. La sincronización en sistemas multimedia, más exactamente, es el aseguramiento de las relaciones temporales entre ULDs del mismo medio y entre ULDs de medios distintos.

Desde un punto de vista más general, podemos clasificar la sincronización tomando en cuanta características del contenido, relaciones temporales y espaciales entre flujos multimedia. Hemos ya explicado ya en el capítulo anterior la diferencia entre medios dependientes del tiempo y los independientes del tiempo, hemos clasificado también los medios como síncronos, asíncronos, isócronos y plesiócronos. Desde este punto de vista, el contenido semántico de la presentación no afecta a la misma.

Un ejemplo de la vida diaria donde podemos observar la sincronización en sistemas multimedia, es la televisión. En este sistema un flujo, el de video, mantiene relaciones temporales entre sus ULDs. Más aún, existen relaciones temporales entre ULDs de los flujos de audio y video. Es bastante notoria una falla en la sincronización en cualquiera de estos dos niveles, por ejemplo, cuando el sonido que escuchamos no corresponde al movimiento de los labios de un comentarista.

El primer paso para la solución de este problema es la especificación de estas relaciones temporales. Aunque este trabajo no trata con los procedimientos necesarios para esta

especificación, el lector puede revisar [6], [8], [9], [11], [20] donde encontrará más información al respecto.

El objetivo de este capítulo es mostrar algunas clasificaciones de sincronización, así como definir de una manera clara y no ambigua los tipos de sincronización que tratará este trabajo. Este capítulo define también algunas propiedades de los tipos de sincronización definidos.

2.6 Criterios de sincronización

Existen, como hemos mencionado, varios criterios de sincronización. Explicaremos primero algunos de estos criterios, para abocarnos después a una sola de estas clasificaciones.

2.6.1 Relaciones en contenidos.

Este tipo de clasificación define relaciones semánticas entre varios medios al momento de producción y reproducción.
Un ejemplo de esto es la relación que existe entre una hoja de cálculo y un gráfico que representa los datos contenidos en la hoja de cálculo. De esta manera, los datos pueden ser representados de dos formas distintas. Otro ejemplo relacionado con el primero, son dos gráficos que dan diferentes interpretaciones de la misma hoja de cálculo.

2.6.2 Relaciones espaciales

Este tipo de sincronización define relaciones espaciales utilizadas para una presentación en un dispositivo de salida en un tiempo dado. Si el medio de salida es bidimensional, como un papel o un monitor de computadora, es necesario definir el espacio bidimensional utilizado por cada medio en cada punto del tiempo. Un ejemplo de esto, es una película subtitulada, en la que los títulos tienen una posición definida sobre los frames de video en cada momento, y esta posición puede cambiar si la especificación lo indica.

2.6.3 Relaciones temporales

Esta clasificación define relaciones temporales entre ULDs de un mismo flujo y/o entre ULDs de flujos distintos. Este tipo de relación se vuelve más interesante, y más complicada de mantener, si al menos uno de estos flujos es dependiente del tiempo. Un ejemplo de este tipo de relaciones es un flujo de video y un flujo de audio que guardan relaciones semánticas.

Esta es la clasificación con la que trabajaremos. Aclararemos ahora algunas clasificaciones de sincronización que podemos tener tomando estas relaciones.

2.7 Sincronización Intraflujo

Esta sincronización se refiere a las relaciones temporales existentes entre ULDs de un mismo flujo. Un ejemplo de esto es la relación que hay entre ULDs de una presentación de video o de audio. Para un video de 25 cuadros por segundo, cada uno de los cuadros debe ser desplegado cada 40ms. La figura 2.11 muestra una secuencia como la mencionada.

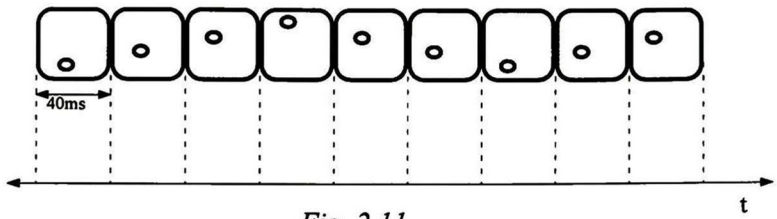


Fig. 2.11
Secuencia de ULDs con sincronización Intraflujo

2.8 Sincronización Interflujo

Este tipo de sincronización se refiere a relaciones temporales entre ULDs de diferentes medios. Nuevamente un ejemplo de esto lo podemos encontrar en una película que integra video, audio y títulos provenientes de distintos medios. Los títulos deben ser presentados durante algunos cuadros de video, además, varios ULDs de audio también deben ser presentados al mismo tiempo que un cuadro de video. Esto lo podemos ver en la figura 2.12.

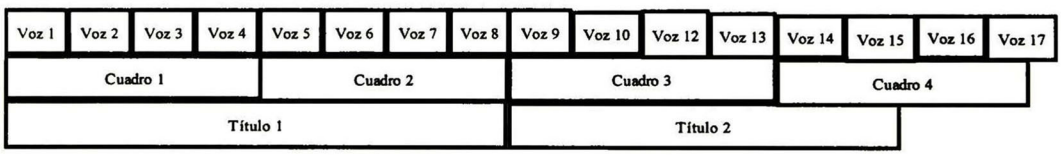


Fig. 2.12
Varios flujos inter-relacionados

2.9 Requerimientos de Presentación

Para la entrega de datos multimedia de manera correcta en la interfaz de usuario, la sincronización es esencial. No es posible proveer una medición objetiva de la sincronización desde el punto de vista de una perspectiva humana. Como la perspectiva humana varía de persona a persona, solamente los criterios heurísticos pueden determinar la claridad de la presentación.

Los requerimientos de presentación comprenden, para la sincronización Intraflujo, la exactitud relativa al retraso en la presentación de los ULDs y, para la sincronización Interflujo, la exactitud en la presentación paralela de los objetos multimedia.

2.9.1 Sincronización sonido-labios (Lip Synchronization)

Se refiere a las relaciones temporales existentes entre un flujo de audio y uno de video para el caso particular del habla humana. Las diferencias temporales entre los ULDs de audio y video son conocidas como el *skew* [3.6.5]. Los flujos que están perfectamente sincronizados, no tienen skew.

El "IBM European Networking Center" realizó algunas mediciones y experimentos sobre lip synchronization. En sus experimentos, los usuarios mencionaban que la presentación estaba fuera de sincronización, pero esto no era molesto. Entonces, los experimentos evaluaron adicionalmente la tolerancia de os usuarios a este tipo de problemas de sincronización [6].

Estos experimentos mostraron que la *sincronización sonido-labios* era más notoria si el cuadro del video estaba más cerca del rostro del comentarista, esto es, los cambios son más notorios mientras mejor se pueda observar los labios de la persona que habla.

Estos estudios fueron hechos en ambientes en los que los experimentadores grabaron la presentación y después la reprodujeron introduciendo desfases artificiales con un equipo de edición profesional. Estos desfases se hicieron cada 40ms, es decir, -120ms, -80ms, -40ms, 0ms, 40ms, 80ms, 120ms. Se escogieron rangos de este tipo debido a la dificultad de la percepción humana para distinguir rangos menores.

El skew negativo fue tomado cuando el video está adelantado del audio. El skew positivo es cuando el video está atrasado con respecto al audio. De estos experimentos se obtuvieron los resultados mostrados en las figuras 2.13 y 2.14.

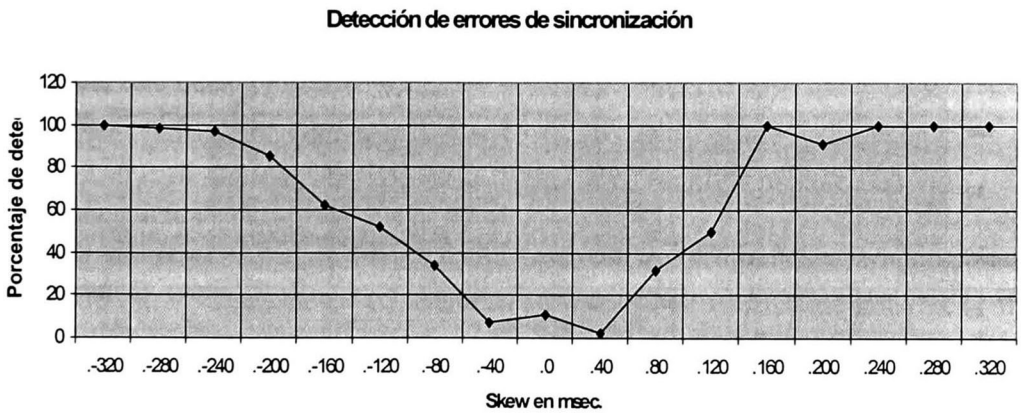


Fig. 2.13

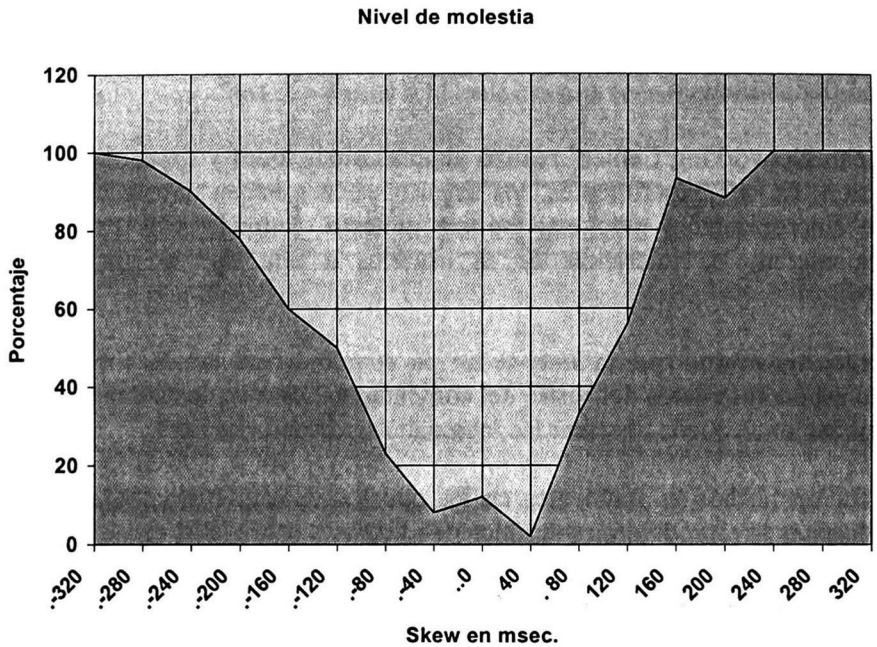


Fig. 2.14

2.9.2 Sincronización imagen-puntero (Pointer Synchronization)

Este tipo de sincronización se refiere a una presentación tipo *slide show*, es decir gráficos presentados en pantalla y explicados por un flujo de audio. Al mismo tiempo, un puntero está señalando las partes del gráfico que el audio explica. Este tipo de sistemas requiere, obviamente, de sincronización entre lo que se explica y lo que se señala o apunta.

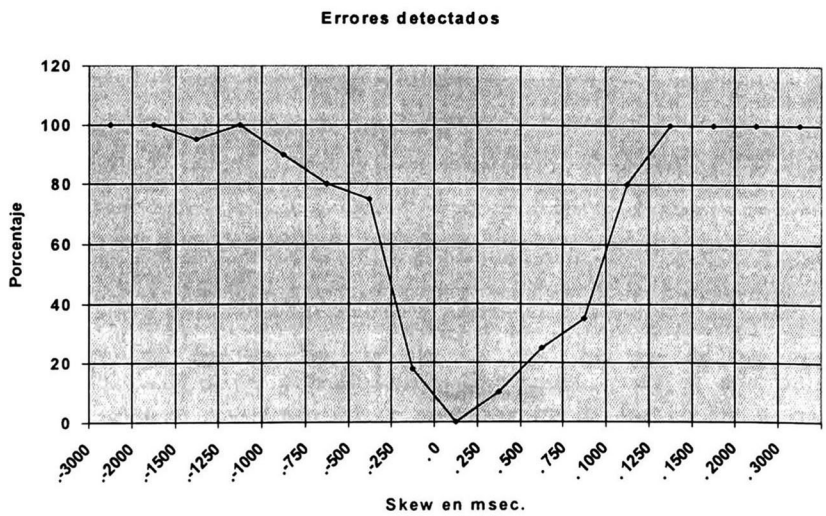


Fig. 2.15

Desde el punto de vista de la percepción humana, la sincronización de puntero es muy diferente de la sincronización de labios, ya que es mucho más difícil detectar el error de sincronización a valores de *skew* cercanos al espacio libre de error [6].

Los resultados obtenidos de este experimento se muestran en las figuras 2.15 y 2.16:

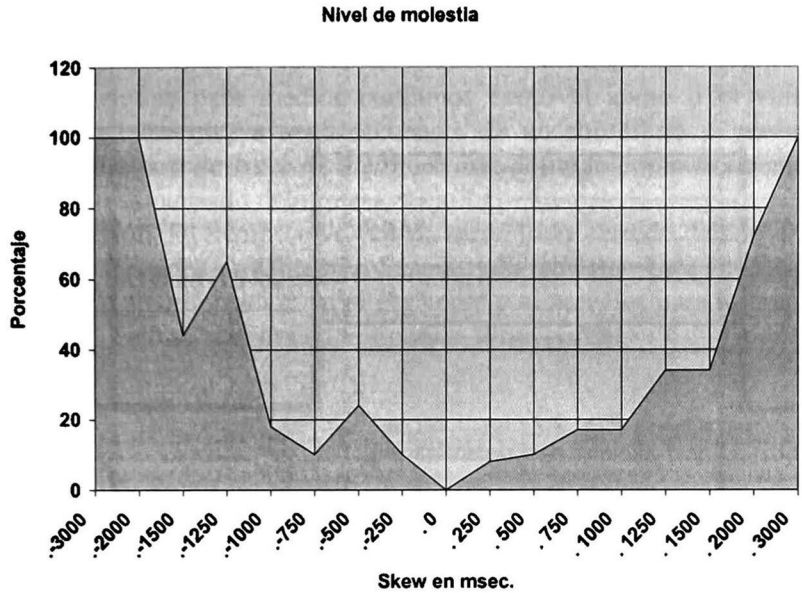


Fig. 2.16

2.10 Diferentes modelos para sincronización en sistemas distribuidos

La sincronización en un sistema distribuido es bastante más compleja que un sistema centralizado, esto debido al almacenamiento distribuido de los objetos distribuidos, retrasos adicionales, posibles pérdidas, entre otros.

2.10.1 Transporte de la especificación de sincronización

En el nodo receptor, el que va a reproducir la presentación multimedia, los componentes que realizan la presentación necesitan los requerimientos de sincronización. Podemos distinguir tres aproximaciones principales para la entrega de esta información:

- *La entrega total de la especificación de sincronización antes del inicio de la presentación*: Esta aproximación es utilizada normalmente para la sincronización sintética (no en vivo). Esta aproximación es simple y permite el manejo de varios nodos como fuente de los objetos. Esta aproximación tiene algunas desventajas, por ejemplo: tiene un retraso inicial inducido por el envío inicial de los datos de sincronización, no funciona en los casos de presentaciones en vivo.

- *Utilización de un canal adicional para la transmisión de la sincronización*: Esta aproximación se muestra en la figura 2.17 y se puede utilizar solamente en el caso que haya un solo nodo fuente. Es útil en presentaciones en vivo ya que, si se utiliza una conexión rápida y confiable, no causa retrasos adicionales ni requiere de espacio extra para su almacenamiento. En el caso de redes no confiables, esta no es una muy buena solución, ya que un pequeño desfase de los datos de sincronización con respecto a los datos de la presentación, pueden ocasionar problemas de sincronización en toda la presentación. Otra desventaja es el requerimiento de un ancho de banda mayor.

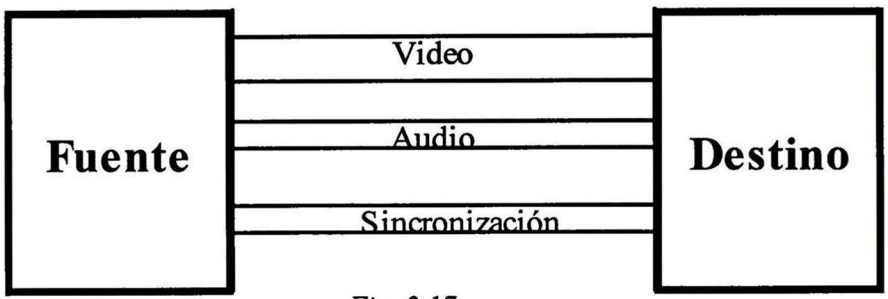


Fig. 2.17
Canal adicional para sincronización

- Multiplexado de flujos. La ventaja de multiplexar varios flujos de datos en un solo canal es que la información de sincronización requerida, es entregada al mismo tiempo que los datos a reproducir. Esto nos evita retrasos adicionales pero mantiene el uso de ancho de banda. Esta aproximación es mostrada en la figura: 2.18. En este trabajo se utiliza esta aproximación para el transporte de datos de sincronización.

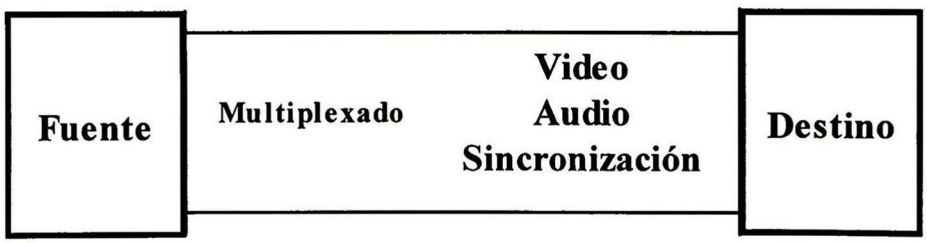


Fig. 2.18
Sincronización multiplexada

2.11 Definiciones sobre sincronización

2.11.1 QoS

QoS es el acrónimo para Quality of Service, Calidad de servicio en español.
La especificación de calidad de servicio para un medio incluye la calidad concerniente a los ULDs individuales de un objeto multimedia, así como a la exactitud con la que las relaciones temporales entre ULDs son cumplidas en el caso de medios dependientes del

tiempo. En el caso de que dos o más objetos multimedia guarden relaciones temporales estrictas, la calidad de servicio se refiere también a la exactitud con que estas restricciones temporales entre ULDs de diferentes medios son cumplidas.
Otros factores que afectan a la calidad de servicio son: retraso en la transmisión (en el caso de elementos geográficamente distantes), skew, jitter, throughput, entre otros. Estos elementos se explican mejor a continuación.

2.11.2 Jitter

El Jitter se define para medios continuos como el audio o el video y resulta de una discontinuidad del medio a consecuencia de un sobreflujo o un bajoflujo (overflow y underflow). La tasa de Jitter de bajoflujo esta definido como la fracción del tiempo que el flujo recibido es pausado debido a un bajoflujo. La tasa de Jitter de sobreflujo está definido como la fracción del tiempo que el flujo recibido es perdido debido al sobreflujo. La suma de estos dos términos es llamada solamente tasa de Jitter. Este Jitter es debido mayormente a las fallas de sincronización entre el cliente y el servidor para la manutención de un flujo continuo [7]. Esto se muestra en la figura 2.19.

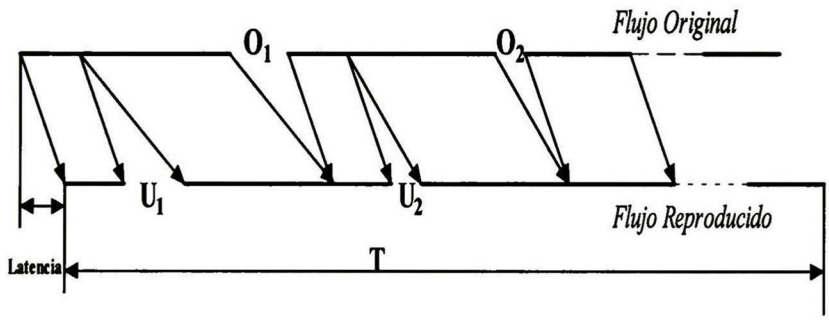


Fig. 2.19
Jitter

La variación en el arribo de paquetes es llamada Jitter [10].

$$\text{Tasa de Jitter de sobreflujo} = \frac{\sum_i O_i}{T}$$

$$\text{Tasa de Jitter de bajoflujo} = \frac{\sum_i U_i}{T}$$

2.11.3 Latencia

La Latencia, definida para transporte de imágenes, audio o video, es el retraso entre la generación de la señal y su reproducción. Esto lo podemos ver en la figura anterior [7].

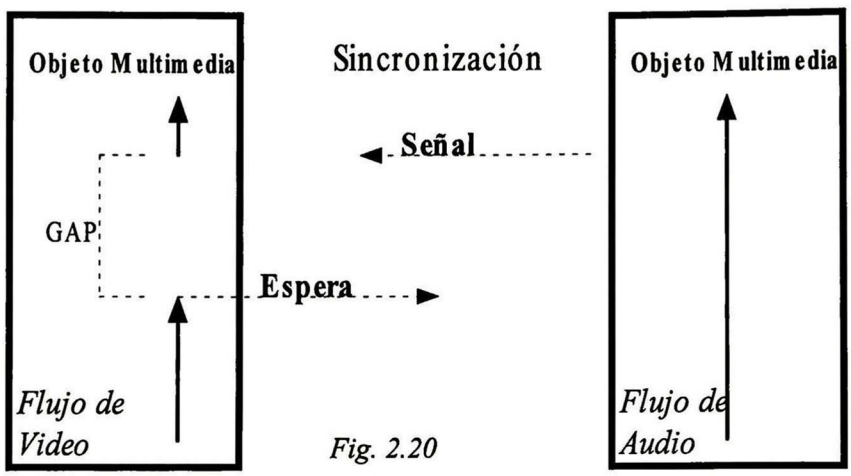


Fig. 2.20
GAP

2.11.4 GAP

El problema del GAP es cuando un objeto multimedia tiene que esperar por otro, es decir, hay un espacio de tiempo en el que un flujo debe esperar a que el otro flujo envíe una señal de sincronización [8]. Esto se ilustra en la figura 2.20.

2.11.5 Buffering

Una solución a algunos problemas mencionados aquí. Consiste en guardar un pequeño buffer de información en el nodo receptor para tener una especie de "colchón" que amortigüe la falta de datos o el retraso de información de la red. La determinación del tamaño óptimo del buffer es un problema clave en el diseño de un sistema multimedia.

2.11.6 Desviación (Skew)

Son las diferencias temporales que hay entre la producción y la reproducción de dos flujos, ie. Si en la producción de una presentación multimedia de dos flujos, digamos x e y , entre dos ULDs, digamos $ULD_{i\text{-}x}$ y $ULD_{j\text{-}y}$, existe una relación temporal de k unidades de tiempo, entonces en la reproducción debe mantenerse esta misma relación. Si la diferencia temporal entre $ULD_{i\text{-}x}$ y $ULD_{j\text{-}y}$ en la reproducción es igual a k, se dice que no hay skew, o que *skew=0*. Si la diferencia temporal entre $ULD_{i\text{-}x}$ y $ULD_{j\text{-}y}$ en la reproducción fuera de $k+m$ unidades de tiempo, diríamos que existe un skew de m unidades de tiempo.

El problema del *skew* se conoce entre el común de la gente como *defasamiento* entre dos objetos multimedia. Esto lo podemos ver mejor en la figura 2.21.

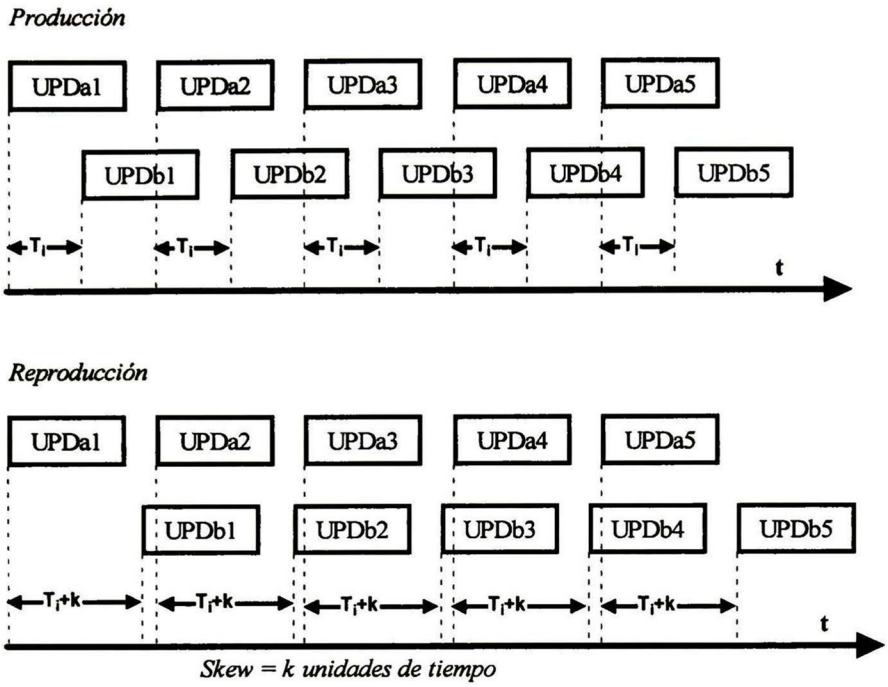


Fig. 2.21
Skew

2.11.7 Razón de error de Bit (Bit Error Rate)

Es la tasa de error en la entrega de información en un sistema distribuido, pero medido en bits por unidad de tiempo.

2.11.8 Razón de error de Paquete (Paquet Error Rate)

Es la tasa de error en la entrega de paquetes en un sistema distribuido.

2.11.9 Razón de velocidad (Speed Ratio)

Es el cociente entre la tasa de presentación actual y el valor nominal de la presentación. En la figura 2.22, para el periodo $t_0 - t_1$, el speed ratio equivale a 6 sobre 6, es decir, uno. Para el intervalo $t_1 - t_2$, el speed ratio es equivalente a 4 sobre 6, es decir, 0.67 [10]. Idealmente, el speed ratio es igual a uno.

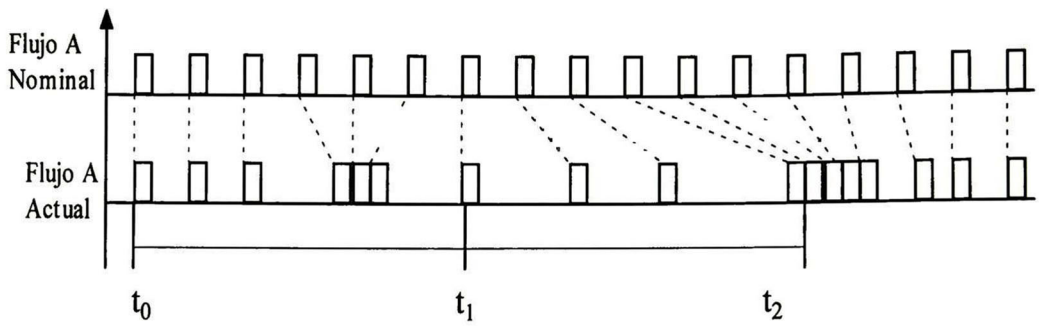


Fig. 2.22
Speed ratio

2.11.10 Razón de utilización (Utilization Ratio)

Este es equivalente a la tasa de presentación actual entre la tasa de entrega disponible. En condiciones ideales, esta tasa es igual a uno.
La duplicación de frames creará una utilización ratio más grande que uno, mientras que el desechar frames causará que sea menor que uno [10].

2.11.11 Retardo

El retardo total de terminal – terminal de un sistema multimedia distribuido consiste en la suma de todos los retardos creados en el sitio fuente, la red y el sitio destino [10].

3

Descripción del problema

3.1 Introducción

Voz sobre la Internet, Voz sobre ATM, voz sobre Frame Relay, hace apenas algunos años, la mayoría de nosotros encontrábamos difícil de imaginar que estas aplicaciones para telecomunicaciones tendrían el auge que tienen hoy en día. Se estima que para el año 2002 cerca del 20% de todas las comunicaciones telefónicas domésticas serán a través de líneas de datos, aunque ahora es solamente el 1% [23].

Las organizaciones alrededor del mundo desean reducir los costos en comunicaciones. La consolidación de las redes de datos y de voz ofrece una oportunidad para bajar costos. La empresa de integrar las redes de voz y las de datos comienza a ser una prioridad para los administradores de las redes. Las organizaciones están proponiendo soluciones que les permitirán tomar ventaja del exceso de capacidad en el ancho de banda en redes de banda amplia para transmisión de voz y datos así como utilizar la Internet y la Intranet de su compañía como alternativas de medios menos costosos [23].

Una gran cantidad de productos y servicios requieren aún de gateways que enlacen las redes de datos y las de voz. Dentro de estos existirá una tecnología capaz de reducir las diferencias causadas por el envío de voz sobre redes de datos que no fueron diseñadas para manejarlas.

El procesamiento de voz necesitará manejar retardos más grandes y variables y cancelar los ecos que serán introducidos del lado del teléfono para que no suene mecánico.

Será necesario también enmascarar los desfases (GAP) causados por el descarte de paquetes durante la congestión.

El lado de procesamiento de paquetes en el gateway tendrá que adaptarse a redes y condiciones variables para asegurar la adecuada conexión entre terminales. También, un entendimiento de cómo se maneja la traducción de configuración de llamadas de diferentes tipos de redes y conexiones es esencial para el manejo de cada llamada.

Una aplicación de voz sobre redes de paquetes debe cumplir con los requerimientos de combinar las redes de datos y las de voz permitiendo tanto a la información de voz como a la de señalización ser transportadas a través de la red de paquetes.

Una gran cantidad de aplicaciones son posibles mediante la transmisión de voz sobre redes de paquetes. Por ejemplo:

3.1.1 Comunicación Intraoficinas

La figura 3.1 muestra la configuración de red de una organización con varias ramas de comunicación en una misma oficina que desea reducir costos y combinar el tráfico para proveer acceso de voz y datos a la oficina principal. Esto se logra mediante la utilización de redes de paquetes para proveer una forma común de transmitir tales medios.

Esta configuración de red sería beneficiada si el tráfico de voz fuera comprimido debido al bajo ancho de banda disponible para estas aplicaciones. Este sistema debe emular las características tanto de un PBX [1]para las terminales de telefonía como de una terminal para el PBX.

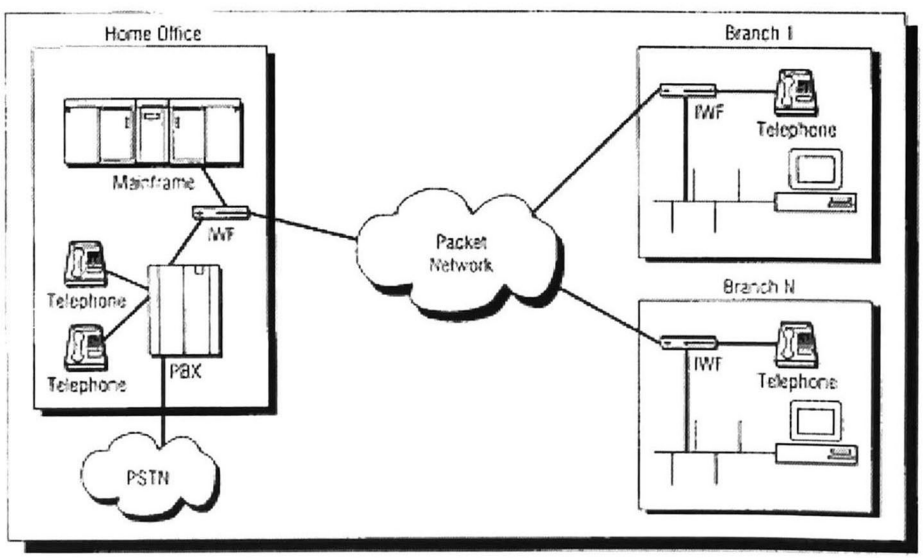


Fig. 3.1
Comunicación Intraoficina

[1] Acrónimo de Private Branch Exchange, una red telefónica privada.

3.1.2 Comunicación Interoficinas.

Una segunda aplicación de la transmisión de voz sobre redes de paquetes es la conexión de oficinas, que no necesariamente se encuentran en el mismo edificio, ni siquiera en la misma región. En este escenario, una organización quiere trafico de voz entre dos localidades sobre la red de paquetes y remplazar las troncales que unen a sus PBXs. Este tipo de aplicación requiere de un ancho de banda mayor al ejemplo anterior.

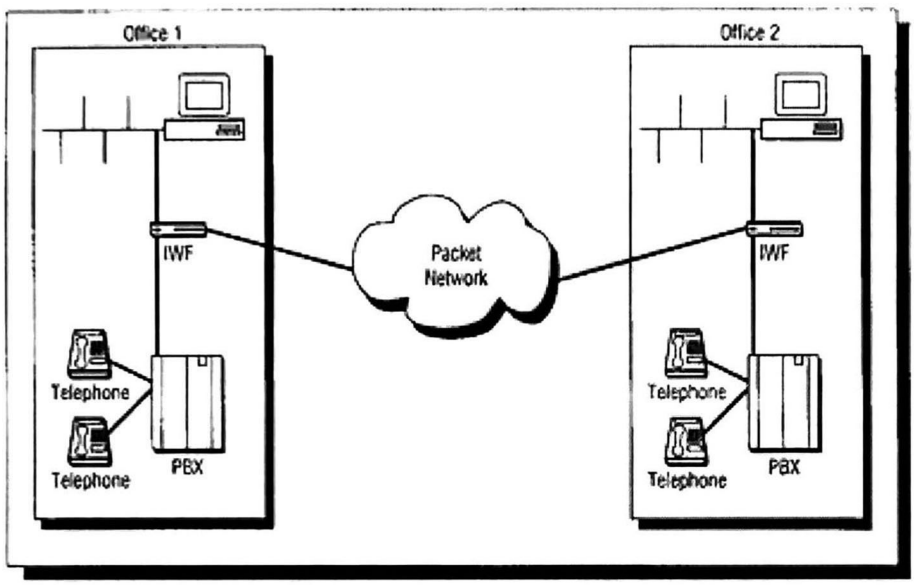


Fig. 3.2

Comunicación Interoficinas

3.1.3 Conexión de una LAN con una PSTN

En este caso, se desea que desde una PSTN se pueda establecer una llamada telefónica con una terminal dentro de una LAN. El caso contrario es también deseable, desde una terminal dentro de una LAN podríamos comunicarnos a un abonado en una PSTN. Esta configuración requiere de una conexión entre PSTN y LAN utilizando un Gateway. Es también necesario agregar características de control y seguridad para evitar el uso no deseado de la red PSTN (fig. 3.3).

3.1.4 Interfaz de PSTN con redes celulares

Otra aplicación de voz sobre redes de paquetes es la interacción redes celulares. Los datos de voz en una red celular digital ya están comprimidos y empaquetados para la transmisión en el aire por el teléfono celular. Las redes de paquetes pueden entonces transmitir los paquetes celulares. En esta configuración se requiere de un dispositivo que traduzca los paquetes de celular a una PSTN (fig. 3.4).

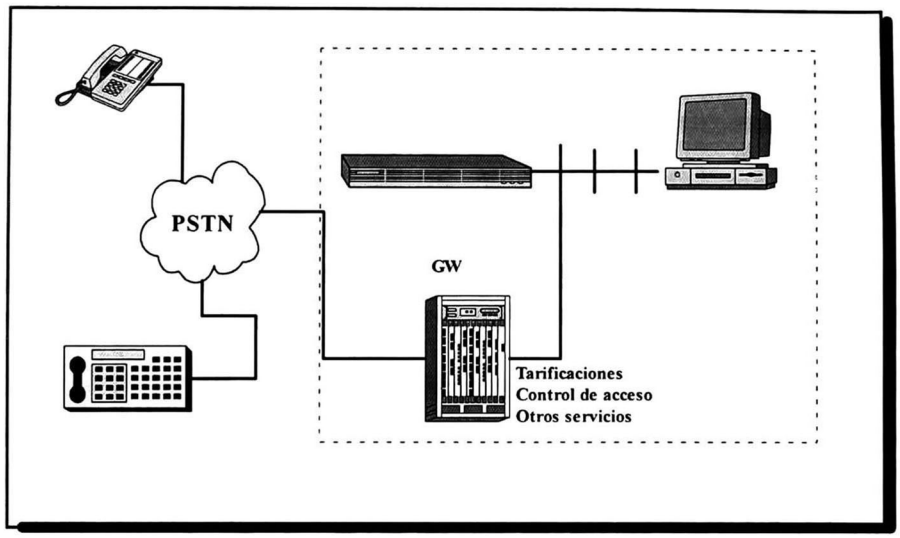


Fig. 3.3
Conexión a PSTN

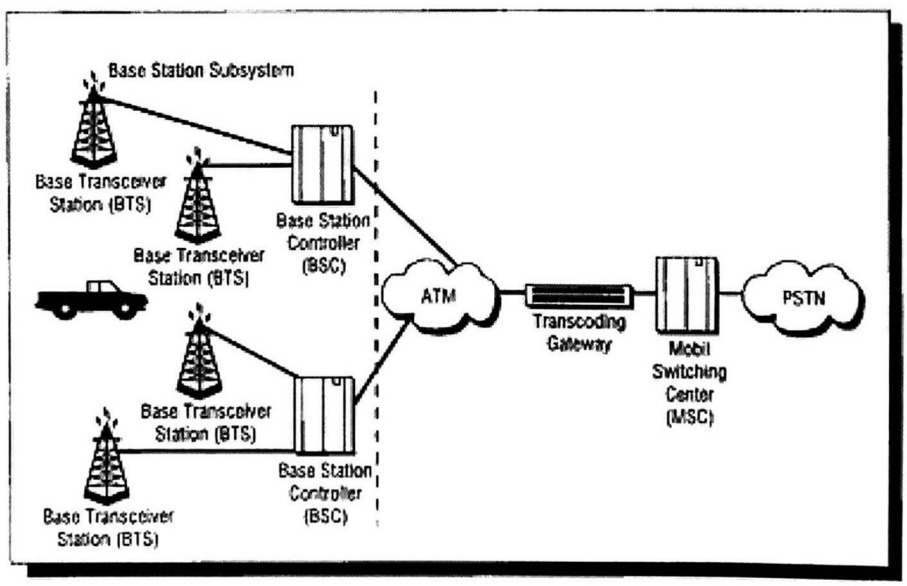


Fig. 3.4
Interfaz a redes celulares

3.1.5 Comunicación By Pass

Este esquema es de los más atractivos. Aquí un abonado a la PSTN desea hacer una llamada de larga distancia a otra PSTN pero utilizando redes de datos en lugar de las troncales entre PSTN's. Un esquema como el descrito se ilustra en la figura 3.5.

Un sistema así trae varios problemas a resolver, por ejemplo: cuando un paquete entra a Internet, no es posible saber la ruta que seguirá o el tipo de redes por las que pasará, esto hace que el tiempo de arribo entre paquetes es independiente entre ellos.

Otros problemas son: retraso impredecible, desorden y pérdida de paquetes. En este esquema es necesario utilizar mecanismos de compresión de voz, sistemas de seguridad y tarificación de la llamada, deben también agregarse capacidades de ordenamiento de paquetes y recuperación ante congestiones.

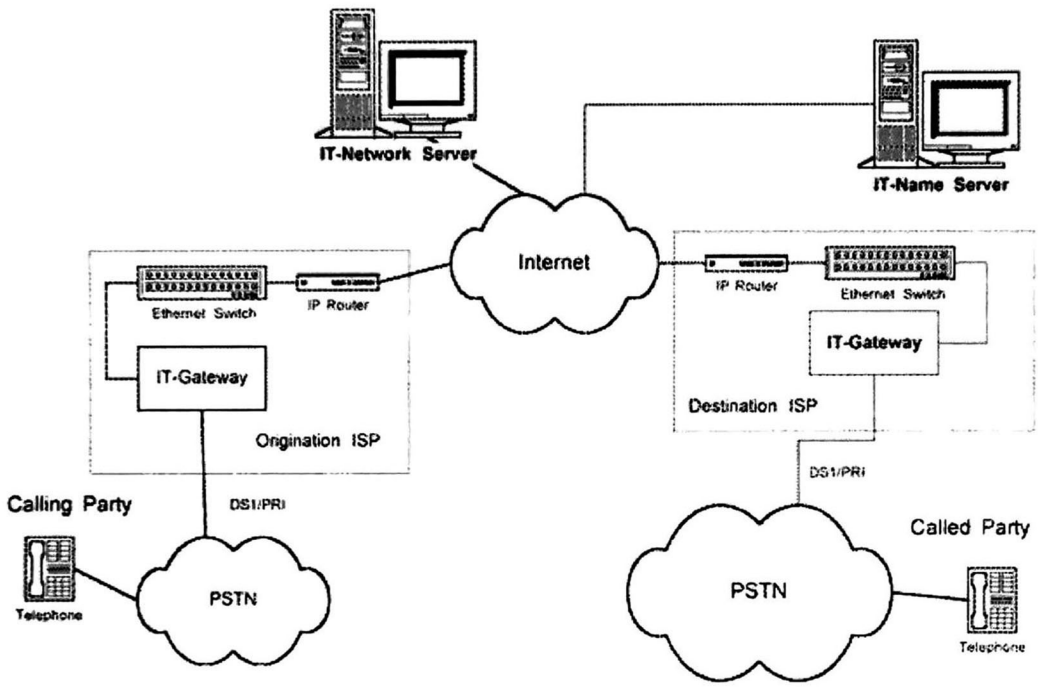


Fig. 3.5
Comunicación By Pass

3.1.6 Calidad de servicio

Las ventajas de la reducción de costo y ahorro de ancho de banda al transportar la voz sobre una red de paquetes, traen consigo algunos problemas de calidad de servicio que son inherentes a las redes de paquetes. Algunos de ellos son: el desorden de los paquetes, el retraso, la pérdida de paquetes, el jitter, el skew y la producción de eco.

3.2 Definición del trabajo

Conjuntando todos estos esquemas obtenemos un sistema bastante completo en el que podemos hacer uso de diferentes servicios, por ejemplo:

- Comunicar dos terminales telefónicas conectadas en una LAN

- Comunicar dos terminales PC con capacidades multimedia conectadas a la misma LAN
- Comunicar una terminal telefónica y una terminal multimedia conectadas a la misma LAN
- Comunicar un abonado de una PSTN a una terminal de una LAN
- Comunicar una terminal en una LAN con un abonado en una PSTN
- Comunicar dos abonados a PSTN utilizando la red de datos como troncal

3.3 Barreras técnicas

El último objetivo de la telefonía IP es, por supuesto, confiabilidad, alta calidad de servicio de voz. Servicios como los que un usuario puede encontrar en cualquier PSTN

En este momento, de cualquier manera, ese nivel de confiabilidad y calidad de sonido no está disponible en la Internet, primeramente por las limitaciones de ancho de banda que permiten la pérdida de paquetes. En comunicaciones de voz, la perdida de paquetes se representa como un desfase (GAP) o periodos de silencio en la conversación, efecto que es insatisfactorio para la mayoría de los usuarios e inaceptable para comunicaciones de negocios [22].

4

Especificación de Requerimientos de Softwarepara el sistema VoIP

4.1 Introducción

Este documento especifica los requerimientos de usuario para el sistema VoIP. Este conjunta los requerimientos globales para el sistema VoIP y las interfaces entre los componentes de hardware y el software del sistema.

El objetivo del sistema VoIP es establecer un marco común de comunicaciones para ambientes de oficina. Este marco está basado en el ambiente LAN comúnmente encontrado en la mayoría de las organizaciones. Se pretende facilitar la migración del equipo actual de telefonía a un sistema de comunicaciones totalmente basado en computadora. El VoIP está orientado a alcanzar total conectividad con equipos basados en el estándar H.323 de la ITU-T.

El mercado global para la telefonía IP podría alcanzar $3.2 billones de dólares para el 2002 [6]. Este producto está siendo desarrollado para reducir costos de llamadas telefónicas de larga distancia y tomar ventaja de la actual red global de computadoras y su reducción en costo. Se ha estimado que la telefonía en Internet será un mercado de $560 millones de dólares para el final de 1999. Para el final de 1997, el tráfico de la telefonía IP alcanzó los 6.3 millones de minutos por mes, de acuerdo a un reporte liberado en mayo de 1998 por consultores de mercado de Frost & sullivan. Eso representa solo el 0.2 por ciento del tráfico total, pero el reporte predice el 10% de todo el tráfico para el 2002.

Notas: Cuando utilicemos la palabra ***debe***, querremos decir obligación, y cuando utilicemos la palabra ***podría*** , querremos decir opción.

4.1.1 Propósito

Este documento fue escrito para diseñadores, ingenieros de pruebas y gente interesada en la materia.
Éste documento debe establecer las bases para la implementación de comunicaciones multimedia, aunque aquí y ahora, solamente especificamos las comunicaciones de voz.

4.1.2 Definiciones, acrónimos y abreviaturas

Host	Cualquier terminal envuelta en una comunicación
Administrador de sistema	Usuario privilegiado, maneja y configura el sistema de VoIP
IP	Internet Protocol
ITU-T	International Telecommunications Union- Telecommunications
LAN	Local Area Network
Partner	Cualquier terminal envuelta en una comunicación
PHNTS	Phone number to the network address Translation Service
PSTN	Public Switched Telephone Network
QoS	Quality of Service
Terminal	Puede ser un Gatekeeper, Gateway, máquina para compresión o máquina para comunicación. La computadora que contiene la tarjeta IP/6 Port FXS y la que contiene la tarjeta IP/4 Trunk card son terminales también.
VoIP	Voice over IP
WAN	Wide Area Network

4.1.3 Panorama general

El sistema es llamado VoIP el concepto es proveer el servicio de telefonía y comunicaciones multimedia sobre redes IP. El sistema utiliza los recursos ya existentes en una organización (PCs multimedia, teléfonos clásicos, máquinas de fax, etc.) ver figura 4.1

El software VoIP sigue la recomendación ITU-T H.323 y es el componente de control de todo el sistema

El software VoIP debe hacer:

- Interactuar con hardware especializado para la PSTN, dispositivos telefónicos y compresión.

- Integrar algoritmos para sincronización intraflujo e interflujo
- Transmitir datos multimedia en redes LAN o WAN via IP.
- Ser compatible con el estándar H.323
- Servicios telefónicos vía IP
- Establecimiento de la llamada
- Monitoreo de la llamada en progreso
- Finalización de la llamada
- Servicios especiales (llmada en espera, bypass, no molestar, correo de voz)

VoIP no hará:

- Asegurar la calidad de servicio
- Asegurar el establecimiento de la llamada (en caso de caida de la red)

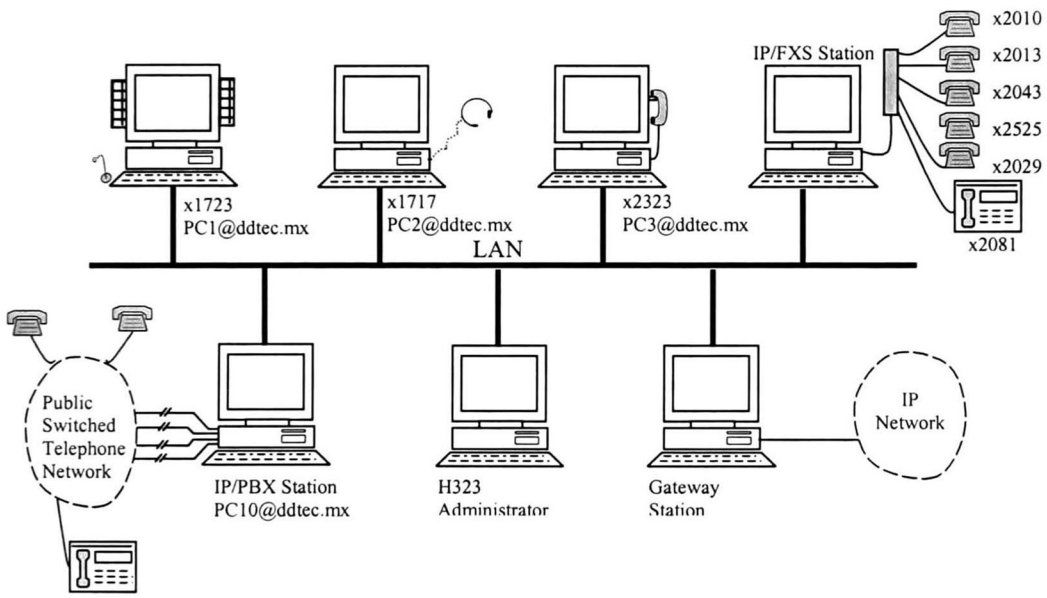


Fig. 4.1 Sistema VoIP

4.1.4 Alcances

Esta sección está organizada de la siguiente manera: la sección 4.2, descripción general, describe aquellos factores que afectarán al producto y su requerimiento; la sección 3 se encuentra en el apéndice C de este trabajo y en idioma inglés. Esto debido a que es un documento que se utilizará para futuros proyectos de investigación y desarrollo.

4.2 Descripción general

El sistema VoIP es un sistema de comunicaciones basado en computadora para ambientes de oficina. Es una extensión del actual servicio telefónico con una clara evolución encaminada a los servicios de videoconferencia integrando facilidades de administración y cuentas.

4.2.1 Perspectivas del producto

Este sistema comprende el siguiente hardware:

- IP FXS card (6port). Interfaz entre 6 teléfonos y el bus PCI de una PC
- IP/4 Port Trunk card. Interfaz entre 4 troncales a la PSTN y el bus PCI de una PC
- IP/ G.723.1. Tarjeta de compresión de voz basada en PCI

El pegamento de estos componentes es el software de control del sistema compatible con H.323. está a cargo de la coordinación entre componentes del sistema, administración, tarifación, seguridad, consistencia y servicios relacionados a la telefonía.

4.2.2 Interfaces VoIP

El ambiente del sistema está mostrado en la figura 4.2. Es independiente de la topología de red, basado en instalaciones de LAN con el requerimiento del uso de IP como protocolo de red.

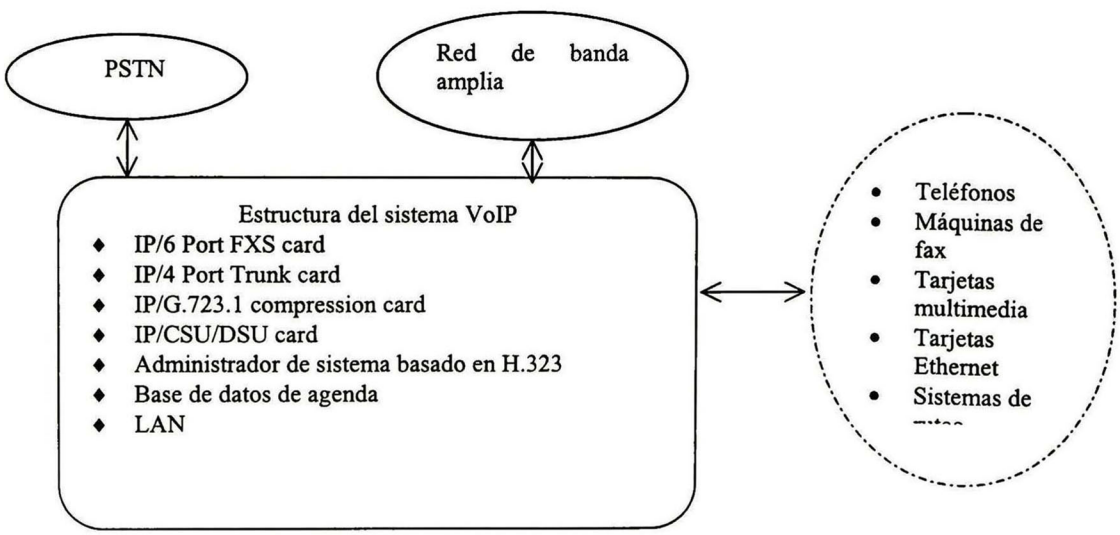


Fig. 4.2 Interfaces VoIP

4.2.2.0 Interfaces de dispositivos

el software central o software de control no interactúa directamente con el hardware. Específicamente, no interactúa con las tarjetas IP FXS, IP Trunk, IP/CSU_DSU and ITU G.723.1 sino con sus Device Drivers.

En el resto de este capítulo, cuando digamos que el Software de Control *debe enviar a o recibir de* alguna de las tarjetas antes mencionadas, querremos decir *enviar o recibir* del Device Driver correspondiente.

Entonces, cualquiera de las tarjetas mencionadas deben interactuar con su Device Driver, el Device Driver deberá interactuar con el Software de Control y el software de control con los Device Drivers y la red IP.

Para mayores detalles en las comunicaciones, ver el capítulo 7.

4. 2.2.1 IP / 4 Port Trunk card

La tarjeta de troncales es responsable de conectar la red de área local con la oficina central telefónica. La tarjeta soporta hasta 4 troncales. La interfaz a la computadora huésped es hecha a través de el bus PCI corriendo a 33 MHz. Para mayor información en el intercambio de mensajes, ver capítulo 7.

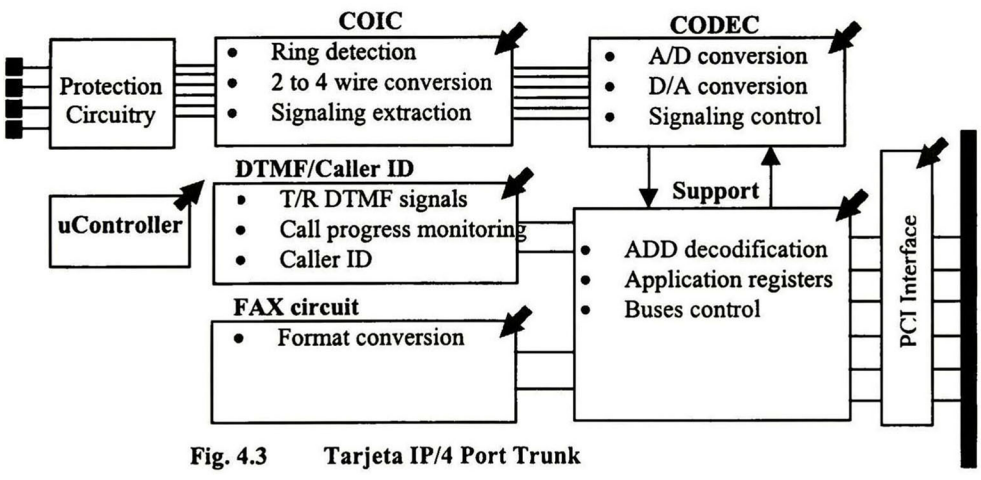


Fig. 4.3 Tarjeta IP/4 Port Trunk

4.2.2.2 IP FXS card (6 port)

La tarjeta FXS es responsable de conectar a los dispositivos tradicionales de teléfono a una red de área local. La tarjeta soporta hasta 6 dispositivos de teléfono. La interfaz a la computadora huésped es hecha a través del bus PCI corriendo a 33 MHz.

Para más información sobre el intercambio de mensajes, ver capítulo 7.

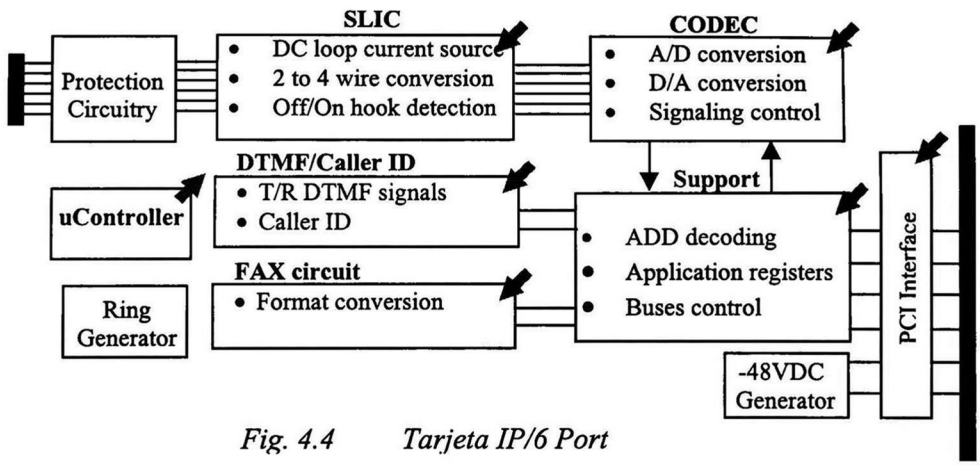


Fig. 4.4 Tarjeta IP/6 Port

4.2.2.3 IP / ITU G.723.1 compression card

La tarjeta de compresión es responsable de comprimir y descomprimir la información de voz que será transmitida a través de un ruteador a la Internet. La tarjeta está basada en un DSP que lleva a cabo estas operaciones. La tarjeta soporta hasta 10 llamadas telefónicas simultáneas. La interfaz ala PC huésped es hecha a través del bus PCI corriendo a 33 MHz.

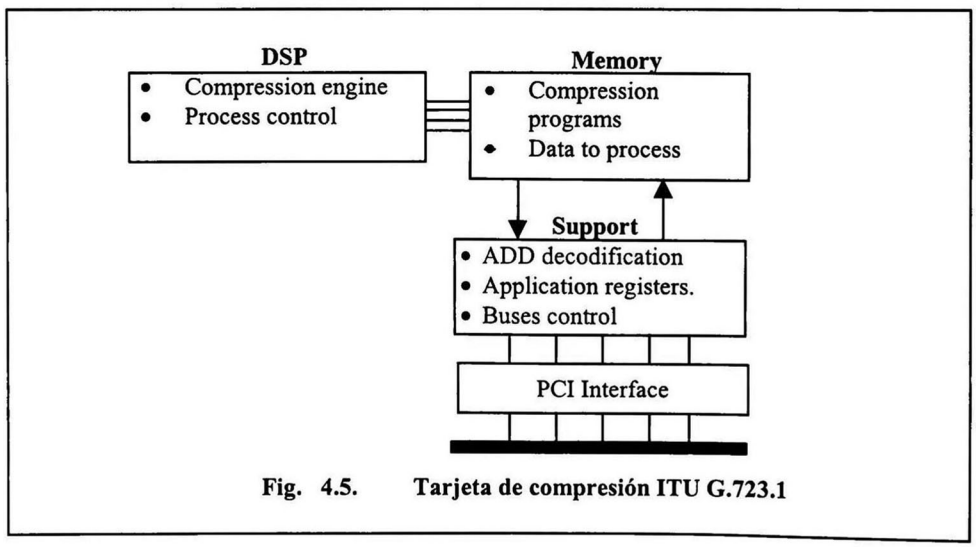


Fig. 4.5. Tarjeta de compresión ITU G.723.1

4.2.2.4 IP / CSU_DSU card

La tarjeta CSU_DSU es responsable de conectar la red de área local a una línea digital T1/E1. El sistema soporta formatosT1/E1 o FT1/FE1. Hasta 24/32 llamadas pueden ser

manejadas por la tarjeta. Canales de voz y datos pueden ser manejados simultáneamente. La interfaz a la computadora huésped es hecha a través del bus PCI corriendo a 33 Mhz.

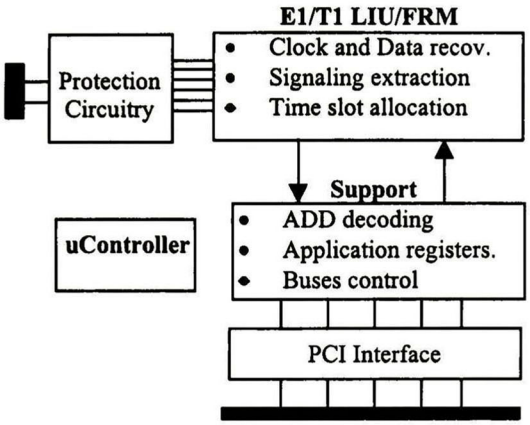


Fig. 4.6 Tarjeta CSU_DSU

4.2.2.5 Administrador H323

El administrador H.323 es un conjunto de programas que hace posible la implementación de servicios sobre la red de datos. El administrador es compatible con la especificación H.323 y es implementado en el Gatekeeper que es un elemento central en H.323 (ver capítulo 5).

- **Definición de usuario**
  - Creación o borrado de cuentas de usuario
  - Modificación de atributos y privilegios a los usuarios
  - Control de acceso a usuarios externos

- **Establecimiento de la llamada**
  - Opera los siguientes servicios telefónicos: intraoficina, interoficina, servicios de telefonía tradicional, Bypass. Ver figuras 4.7, 4.8, 4.9 y 4.10
  - Negociación de los recursos a ser utilizados por el usuario
  - Control de señalización de llamada
  - Conversión de números telefónicos a su correspondiente dirección IP y viceversa
  - Identificación de llamada
  - Conmutación automática a una extensión telefónica
  - Sincronización de comunicación de múltiples medios de la misma fuente al mismo destino
  - Restricción de servicios y números telefónicos
  - Establecimiento de conferencia

- **Monitoréo de llamada en progreso**
  - Estado de la llamada
  - Almacenamiento de las estadísticas de llamada de usuario (número, duración y fecha)
  - Servicio de tarifación

- **Finalización de la llamada**
  - Liberación de las fuentes utilizadas por el usuario

- **Servicios especiales**
  - Directorio telefónico
  - Base de datos de Agenda
  - Correo de voz almacenado en la terminal destino
  - Servicios de seguridad

4.2.2.6 Base de datos de agenda

Sobre la identificación del usuario a ser llamado o la identificación del número telefónico de la llamada entrante, el sistema debe desplegar en la pantalla la información disponoble acerca de la persona que llama o a ser llamada.

La base de datos debería seguir las siguientes reglas.

- Adaptado a la aplicación (centro de cuiddo médico, banco, agencia de viajes, agencias de gobierno, etc.)
- Los datos a ser desplegados pueden seguir cualquier formato definido por usuario (de un conjunto definido de formatos)
- Los datos pueden ser creados, editados o actualizados durante la llamada
- La información se almacena en bases de datos locales
- Los datos pueden ser actualizados a/desde la base de datos general

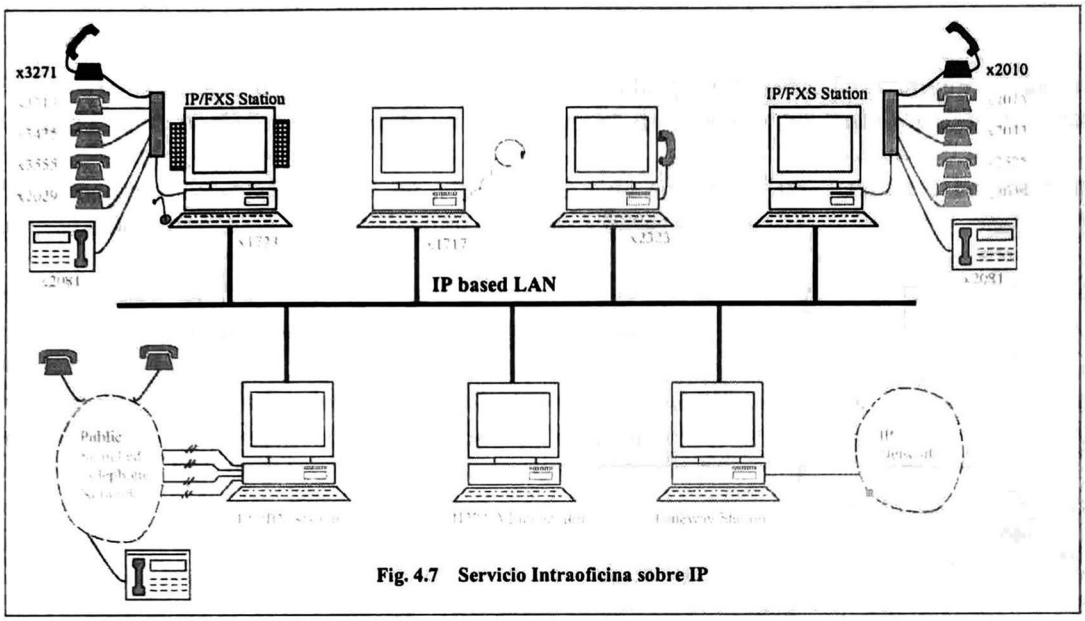


Fig. 4.7 Servicio Intraoficina sobre IP

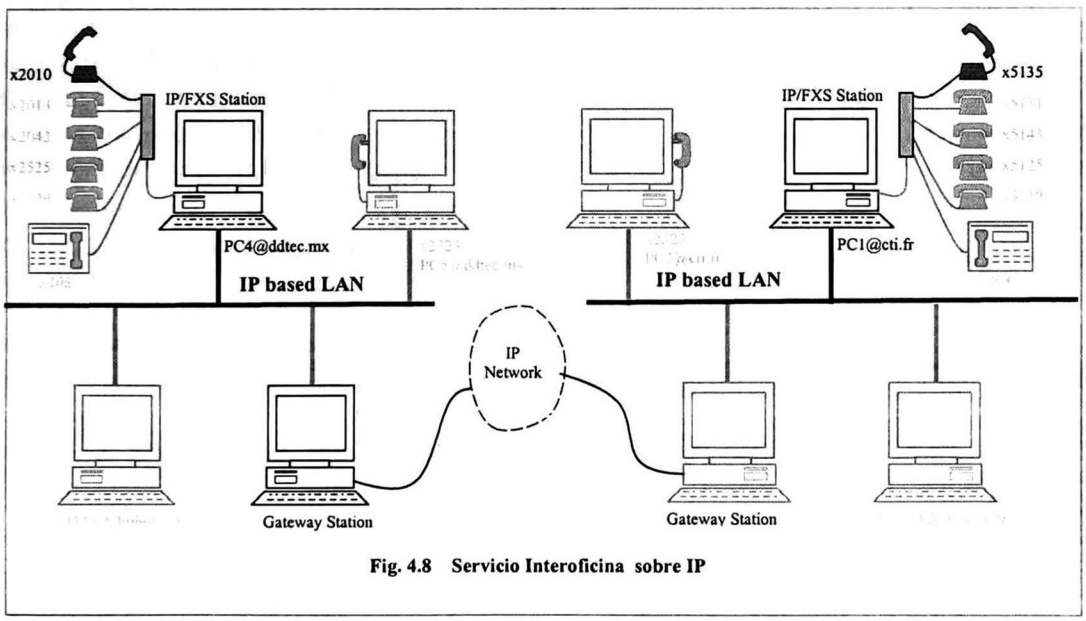


Fig. 4.8 Servicio Interoficina sobre IP

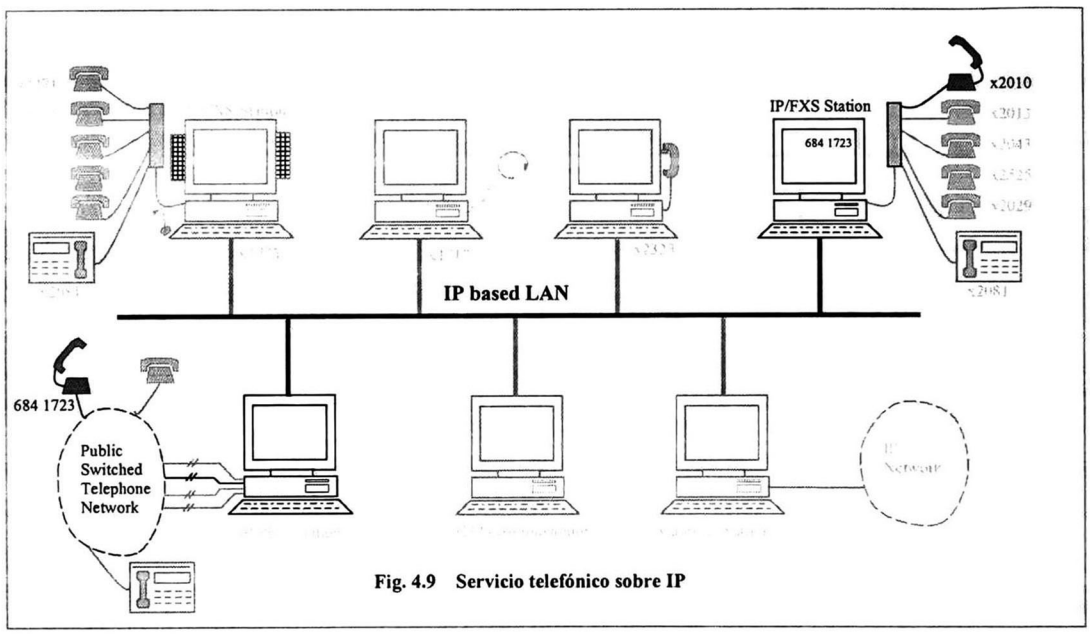


Fig. 4.9 Servicio telefónico sobre IP

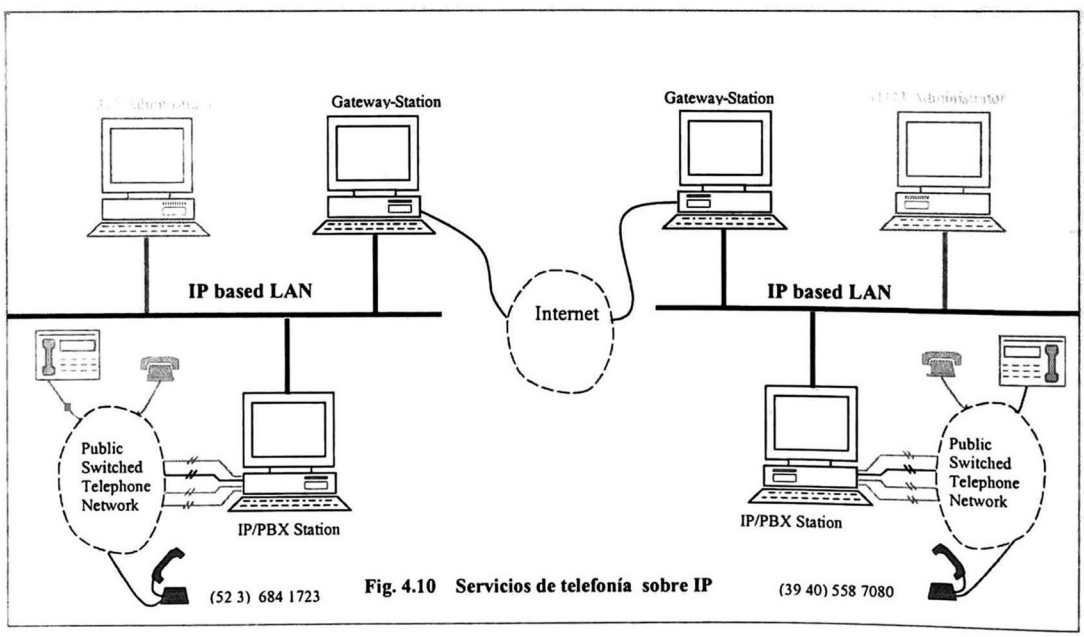


Fig. 4.10 Servicios de telefonía sobre IP

4.2.3 Características de usuario

Los usuarios no son especialistas o gente con una alta preparación. De cualquier manera, los usuarios conectados a través de un computadora dedicada necesitarán un conocimiento mínimo de computación.
El sistema requiere también de un administrador de sistema. Esta persona debe tener un amplio conocimiento en administración de redes y bases de datos

4.2.4 Restricciones generales

El sistema estará limitado por las siguientes políticas:

- Debe ser compatible con H.323 y sus estándares asociados
- Debe trabajar sobre redes IP, haciendo transparente la red subyacente
- Serán tomados tres niveles de seguridad
  Nivel de cuenta: proveer una política basada en costos
  Nivel de administración: proveer restricciones geográficas en ambos sentidos para limitar las comunicaciones con un sitio definido
  Nivel seguro: proveer servicios de compresión y encipción
  Es posible tener operaciones mezcladas de estos tres niveles
- Debe existir una interfaz con la PSTN
- Puede ser accesado a través de equipo H.323 dedicado

4.2.5 Prerrequisitos y dependencias

Se asume la existencia de conexiones con la LAN y la PSTN. Es deseable una conexión con Internet.
Se requiere el siguiente equipo:

- Una tarjeta de interfaz a la red
- Una conexión a la red IP
- Micrófono y bocinas (opcionales)
- Microprocesador péntium, equivalente o superior
- Un mínimo de32 MB de RAM
- Almacenamiento principal: 64MB

4.2.6 Perspectivas futuras

El sistema VoIP será desarrollado en las siguientes etapas:

a) En la primera fase será considerado solamente en ambientes LAN. Solo serán considerados los estándares relacionados con H.323 que estén involucrados con ambientes locales. La comunicación es establecida utilizando IP, haciendo independiente el sistema para cualquier topología en particular
b) En la segunda fase, la comunicación con la Internet será la meta principal. Esta interconectividad envolverá aspectos de seguridad, sincronización multimedia, calidad de servicio e interconexión con la PSTN.
c) En la tercera fase, será implementada la videoconferencia multipunto IP con un inicial manejo de documentos (acceso, almacenamiento de la conferencia, ...)

5

PSTN y H.323

Antes de hablar de la implementación de la Red de Servicios Digitales Integrados, los esquemas de instalación de esta nueva tecnología, y de explicar los protocolos que utiliza (necesarios para controlar dispositivos RDSI), sería adecuado explicar brevemente como se ha ido desarrollando el concepto de RDSI, y como surgió la necesidad su implantación en el mundo entero.

Por lo tanto, vamos a ver a modo de breve introducción como se desarrollaron los acontecimientos, hasta llegar a nuestros días (en lo que a comunicaciones se refiere).

Años 60: Se encuentra la solución a un viejo problema, la pérdida de calidad de sonido en las llamadas a larga distancia. La solución consistía en utilizar canales de larga distancia digitales; en estos canales la voz era digitalizada y enviada como datos numéricos, volviéndola a convertir en una señal analógica en el otro extremo de la línea.

Puesto que en los enlaces digitales la información no sufre deterioro, las llamadas continentales podían tener la misma calidad de sonido que las llamadas locales. El esquema de digitalización elegido fue tomar muestras que eran de 8 bits a una velocidad de 8000 muestras por segundo; esto significaba que estos canales debían funcionar a 64000 bits por segundo (8 bits * 8000 muestras).

Años 70: Las compañías telefónicas se enfrentan a un nuevo desafío; las grandes empresas están interesadas en poder interconectar sus ordenadores; para satisfacer esta nueva demanda se crean las primeras redes experimentales de transmisión de datos.

Año 1984: Asamblea general de la CCITT. Este organismo, dependiente de la ONU, tiene como función establecer los estándares técnicos utilizados en telefonía, con el fin de garantizar la compatibilidad entre los equipos de las diferentes compañías. En esta reunión se habla de los canales digitales, del imparable aumento de las comunicaciones por ordenador y de las nuevas demandas ya aparecidas o de previsible aparición (fax, videotexto, videoconferencia, televisión por cable,... (todas ellas las explicaremos en este trabajo)), y se toma una decisión histórica: la red telefónica mundial deberá reconvertirse en una red de transmisión de datos. El plan es que, en el siglo XXI, las típicas líneas analógicas utilizadas por los teléfonos de voz se habrán sustituido por líneas digitales capaces de ofrecer cualquier tipo de servicio, inventando o por inventar; esta nueva red se bautiza con el nombre de RDSI (Red Digital de Servicios Integrados).

La idea era muy buena, pero presentaba un problema enorme, la construcción de esta red. Si se quería que el proyecto fuera viable, la nueva RDSI debía crearse a partir de la vieja red de voz.

El esquema finalmente elegido fue el de un desarrollo en dos fases; en una primera fase se sustituirían las viejas centrales de relees por nuevas centrales computerizadas, que aunque serían compatibles con los sistemas antiguos podrían ofrecer los servicios requeridos por la nueva red; paralelamente, todos los canales de comunicación (no solo los de larga distancia) se irían reconvirtiendo en canales digitales.

Esto permitiría la existencia de un período de transición durante el cual estarían entremezclados enlaces analógicos y digitales y que concluiría en la RDI (Red Digital Integrada), una red en la que el único enlace analógico sería el que une el teléfono del abonado con la central. Llegados a este punto, se entraría en la segunda fase, que consistiría en alargar los enlaces digitales hasta los abonados; la RDSI habría nacido.

Años 90: Muchos países han completado la construcción de la RDI; puede ponerse en marcha la RDSI. Esta es la situación actual: como puede verse en la figura 3, en el contexto de la RDI el teléfono del abonado está conectado a un convertidor analógico/digital que convierte la señal eléctrica en información binaria que será transmitida a través de un canal de datos; en el otro extremo del canal, un convertidor digital/analógico reconstruye la señal original.

No olvidemos que en la red telefónica, el canal de voz es la unidad básica de funcionamiento; esto significa que la RDI estará formada por grupos de canales de 64 kbps. En Europa y 56 kbps. En EE.UU., lo que también supone que esta deberá ser la velocidad de los canales RDSI.

Actualmente no se puede decir que las redes digitales estén en todo el mundo, por este motivo, es necesario utilizar los recursos con los que por ahora contamos.

5.1 PSTN (Public Switched Telephone Network)

La Red Telefónica Pública Conmutada

Un cable tendido entre dos computadoras puede transferir datos a velocidades de memoria, por lo común, 10^7 a 10^8 bps. La tasa de errores generalmente es tan baja que es difícil medirla, normalmente de un error por día. Esto equivale a un error por cada 10^{12} o 10^{13} bits enviados.
En contraste, la línea telefónica transmite alrededor de 10^4 bps, y tiene una tasa de error de 1 por cada 10^5 bits enviados.

Cuando Alexander Graham Bell patentó el teléfono en 1876, había una demanda enorme por su invento. El mercado inicial fue para la venta de teléfonos, que venían por pares, era obligación del cliente tender un alambre entre ellos. Si el propietario de un teléfono quería hablar con otros n propietarios de teléfonos, debía tender alambres individuales a cada una de las n casas. En el lapso de un año, las ciudades estaban cubiertas de alambres que pasaban sobre las casas y los árboles en una horrible maraña [16].

Bell formó la Bell Telephone Company, que abrió su primera oficina de conmutación en 1878 en New Haven, Conneiticut.

La compañía tendió un cable a la casa u oficina de cada cliente. Para hacer una llamada, el usuario debía dar vuelta a una manivela en el teléfono, con lo que producía un timbrazo en la oficina de la compañía de teléfonos para atraer la atención de un operador, que a continuación conectaba en forma manual a quien llamaba con quien era llamado. Pronto, surgieron por todas partes oficinas de conmutación de Bell Systems y la gente quiso hacer llamadas de larga distancia.

Primero conectaron todas las oficinas entre sí, pero era un problema, entonces decidieron crear oficinas de conmutación de segundo nivel. Esta jerarquía creció hasta cinco niveles. Hoy en día los sistemas telefónicos se organizan de manera muy similar, tienen una organización jerárquica altamente redundante. Cada teléfono se comunica, mediante un par de alambres de cobre, a la oficina central local más cercana, esta distancia puede ser de 1 a 10 Km.

La concatenación del código de área y los tres primeros dígitos del número telefónico especifican de manera única una oficina central. La conexión entre el abonado (usuario final) y la oficina central local, se conoce como lazo local. Si los lazos locales instalados actualmente se unieran y se extendieran a la luna, darían una vuelta redonda unas 1000 veces [16].

Si un abonado llama a otro que se encuentra en la misma oficina central local, s establece una conexión eléctrica directa y se mantiene mientras dure la llamada. Si el teléfono al que llama está conectado a otra oficina final, esta oficina final se comunica a una oficina de

conmutación llamada oficina de cargo o tándem. Estas líneas de comunicación se llaman troncales de conexión con cargo.

Más arriba en la jerarquía, se encentaran las oficinas primarias, las seccionales y regionales que se comunican entre sí mediante troncales de intercargo de un gran ancho de banda.

Actualmente se utiliza la señalización digital, un uno o un cero lógicos. La ventaja de este tipo de señalización, es que puede pasar por un número arbitrario de regeneradores de señal, sin que se afecte el mensaje original. Otra ventaja, es la posibilidad de mezclar diferentes medios (audio, voz, video, datos, etc.).

Actualmente, las conexiones de lazos de larga distancia son digitales, pero los lazos locales siguen siendo analógicos.
Debido a esto, cuando deseamos hacen una transmisión desde nuestras computadoras por medio de una línea telefónica, primero se deben convertir los datos digitales de nuestra computadora a datos analógicos que pasen por los lazos locales, después, estos datos deben convertirse a digitales para transmitirse entre terminales, de esto se convierten en datos analógicos que pasan por el lazo local del otro host, el módem que tiene el otro host convierte estos datos analógicos a digitales para entrar en el otro computador [16].

5.2 Recomendación ITU H.323

5.2.1 Introducción.

La posibilidad de comunicaciones de voz viajando por Internet en vez de la PSTN se hizo realidad por primera vez en Febrero de 1995 cuando Vocaltec, Inc. Introdujo su Internet Phone Software. Diseñado para correr en una 486/33MHz (o superior) equipado con una tarjeta de sonido bocinas micrófono y módem, el software comprimía la señal de voz y la traducía en paquetes IP para su transmisión en Internet. Este software de comunicación PC-PC trabaja solamente si las dos partes está utilizando el mismo software [22].

Dentro de los próximos años, la industria incrementará el ancho de banda de las Backbone actuales de las redes de cómputo por la adopción de redes ATM (Asynchronous Transfer Mode), diseñadas para manejar flujos de voz, video y datos. Tal optimización de la red será un gran paso adelante para eliminar la congestión de la red y la pérdida de paquetes asociada con esta congestión. La industria de Internet está también atacando el problema de la confiabilidad y la calidad de sonido sobre Internet a través de la adopción gradual de estándares. El esfuerzo de establecer estándares se enfoca en los tres elementos centrales de la telefonía en Internet: el formato de codificación de audio, protocolos de transporte y servicios de directorio.

En mayo de 1996, la Unión Internacional de Telecomunicaciones (ITU) ratificó la especificación H.323, la cual define como será transportado el tráfico de voz, video y datos sobre redes de área local basadas en IP; también incorpora el estándar T.120 de conferencia de datos. H.323 está basada en el "protocolo de tiempo real / protocolo de control de tiempo real" (RTP/RTCP) para manejo de señales de audio y video [22].

5.2.2 Especificación

El estándar H.323 provee un fundamento para la comunicación de audio, video y datos a través de redes basadas en IP, incluyendo la Internet. Como objetivo del estándar, las aplicaciones y productos multimedia de diferentes proveedores pueden interoperar, permitiendo a los usuarios la comunicación sin preocuparse de la compatibilidad.

H.323 es una recomendación *sombrilla*[2] de la Unión Internacional de Telecomunicaciones (International Telecommunication Union ITU) que establece estándares para comunicaciones multimedia sobre redes de área local (LANs) que no provee una calidad de servicio (QoS) garantizada. Podemos encontrar este tipo de redes actualmente en cualquier empresa e incluyen las siguientes tecnologías de red: TCP/IP e IPX sobre Ethernet, Fast Ethernet y Token Ring [21].

La especificación H.323 fue aprobada en 1996 por el grupo de trabajo 16 de la ITU. Una segunda versión fue aprobada en enero de 1998. El estándar es amplio en metas e incluye tanto dispositivos *stand alone* como computadoras personales y conferencias punto a punto y multipunto.

H.323 es parte de una larga serie de estándares de comunicaciones que permiten videoconferencia a través de un rango amplio de redes. Conocido como H.32X, esta serie incluye H.320 y H.324 que tratan con las comunicaciones entre ISDN y PSTN (H.323 [37] también maneja el control de la llamada, manejo de multimedia, manejo de ancho de banda e interfaces entre diferentes tipos de redes).

Esta recomendación puede ser utilizada tanto para comunicaciones de solo voz o para multimedia completa. Se espera que H.323 tenga más aceptación conforme pasa el tiempo debido a que:

- H.323 define estándares de multimedia para la infraestructura existente (redes de tipo IP). Diseñado para compensar los efectos de las latencias altamente variables en las redes LAN, H.323 permite el uso de aplicaciones multimedia sin cambio de la infraestructura de red.
- Las redes IP son cada vez más aceptadas. El ancho de banda de Ethernet de 10Mbps a 100Mbps y Gigabit lo hace uno de los productos más atractivos en el mercado.
- H.323 provee interoperabilidad dispositivo a dispositivo, aplicación a aplicación y proveedor a proveedor, permitirá interactuar a productos de diferentes marcas.

El objetivo de H.323 es simple, proveer una protocolo común tal que los productos de software para comunicación de diferentes fabricantes puedan trabajar bien juntos.

H3.23 describe terminales(las terminales pueden ser PCs, o dispositivos standalone.) y servicios para multimedia sobre redes locales que no garantizan la calidad de servicio (QoS). Los flujos pueden ser de voz, datos, video o cualquier combinación.

---

[2] Es decir que cubre todos los aspectos relativos a Multimedia sobre IP

Las redes locales sobre las cuales pueden trabajar las terminales H.323 pueden ser un solo segmento o múltiples segmentos con topologías muy complejas.

La transmisión de datos de voz es obligatoria mientras que los de datos y video son opcionales.

5.2.2.1 Las aplicaciones.

- Videoconferencia
- Telefonía en Internet y videotelefonía.
- Computación colaborativa.
- Juegos en red.
- Buisness Conference Calling (BCC)
- Aprendizaje a distancia.
- Soporte y ayuda para aplicaciones de escritorio.
- Compras interactivas.
- Etc.

5.2.2.2 Beneficios

- Estándares de Codecs.
- Interoperabilidad
- Independencia de redes.
- Independencia de plataformas y aplicaciones.
- Soporte multipunto.
- Soporte multicast.
- Manejo de ancho de banda.
- Conferencia inter-redes.

5.2.2.3 Arquitectura.

H.323 cubre los requerimientos técnicos para comunicaciones de audio, video, voz y datos en redes LAN sin garantizar la calidad del servicio. H.323 no incluye el manejo de las capas de transporte o física de una LAN, solamente los elementos necesarios para la interacción con la Switched Cirtcuit Network (SCN).

Existen cuatro componentes principales: Terminales, Gateways, Gatekeepers, y Unidades de Control Multipunto (MCU). La figura 5.1 ilustra al sistema y sus componentes.

5.2.2.4 Terminales.

Son los puntos finales de una LAN y los medios de comunicación entre usuarios. Estas proveen comunicaciones en tiempo real en ambos sentidos. La figura 5.2 ilustra los componentes de una terminal.

Todas las terminales deben soportar voz, el video y los datos son opcionales. El módulo *Receive Path Delay* es opcional para los flujos de audio y video, este puede ser utilizado para proveer *lip synchronization* y/o control de jitter. H.323 especifica el modo de operación requerido para que las diferentes terminales de audio video y/o datos puedan trabajar en conjunto. Este es el estándar dominante para la próxima generación de teléfonos Internet, terminales de audio conferencia y video conferencia.

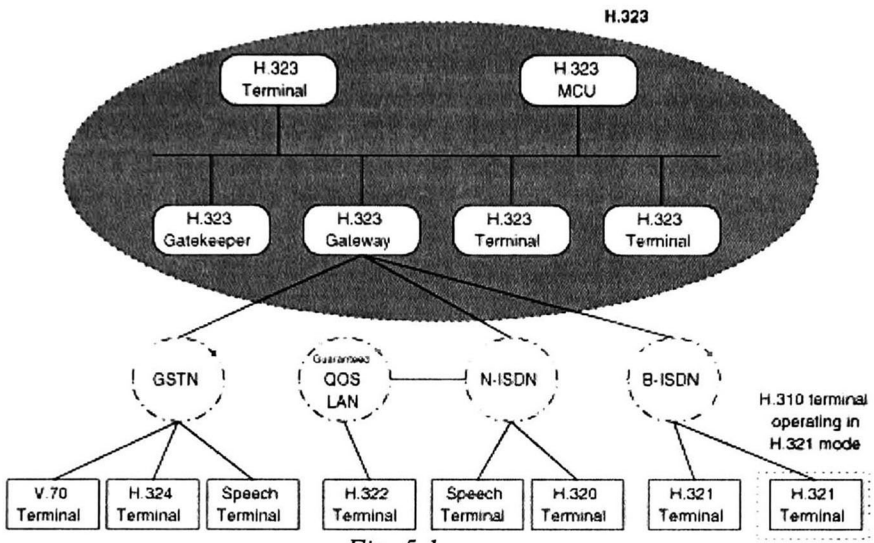


Fig. 5.1
Terminales H.323

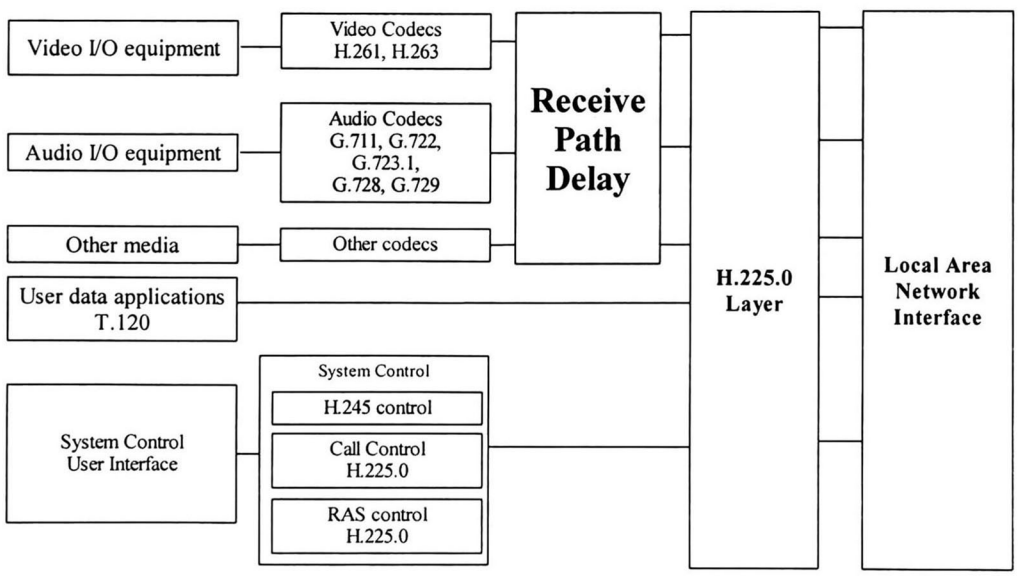


Fig. 5.2
Configuración de componenetes H.323 en módulos

Todas las terminales deben soportar H.245 [39] para la negociación del canal e intercambio de capacidades. Q.931[46] para la señalización de las llamadas y configuración de la llamada. RAS (Registration/Admission/Status) es utilizado para establecer comunicación con los gatekeepers. RTP/RTCP es el protocolo que para la secuenciación de los paquetes de audio y video.

5.2.2.5 Gateways.

El Gateway es un elemento opcional en una conferencia H.323. Estos proveen servicios de traducción entre terminales (H.225.0 [38] a H.221; H.245[39] a H.242), traduce entre Codecs de audio y video. En la figura 5.3 se muestra un gateway H.323/PSTN (Public Switched Telephone Network)

En general, el propósito de un gateway es reflejar las características de un endpoint de una LAN a un endpoint de la SCN y viceversa. Un gateway tiene por función:

- Establecer ligas con terminales PSTN.
- Establecer ligas con terminales H.320 [44] sobre ISDN.
- Establecer ligas con terminales H.324 [45] sobre redes PSTN.

Son dejadas al diseñador las siguientes funciones:

- El número de terminales H.323 que se comunican a través del gateway.
- El número de conexiones SCN.
- El número de conferencias simultaneas soportadas.
- Las funciones de conversión entre flujos de audio, video y datos.
- Funciones de multipunto.

Un gateway no es requerido si las conexiones con otras redes no son necesarias, dado que las terminales podrían comunicarse directamente con cualquier otra.

Las terminales se comunican con los Gateways por medio de los protocolos H.245 y Q.931

Con los transcoders apropiados, un gateway H.323 podría soportar terminales con H.310, H.321, H.322 y V.70.

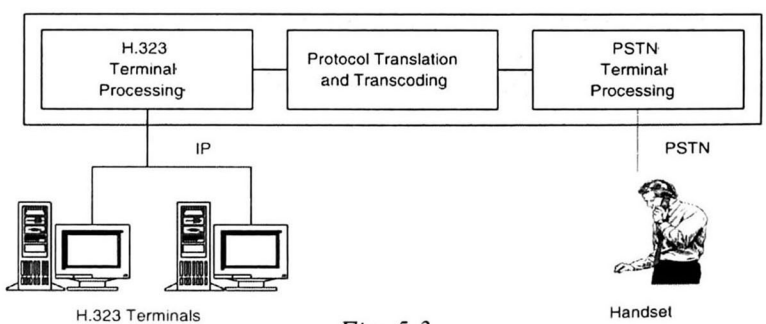


Fig. 5.3
Comunicación a PSTN

5.2.2.6 Gatekeepers.

Un gatekeeper es el componente más importante en un sistema H.323. actúa como el punto central para todas las llamadas dentro de su zona y provee servicios de control de llamada a los puntos terminales que tiene registrados.

El Gatekeeper realizan dos funciones muy importantes para el control de llamadas:

- Traducción de direcciones de aliases de la LAN para terminales y Gateways a direcciones IP o IPX, esto especificado en el protocolo RAS.
- Manejo del ancho de banda para comunicaciones simultaneas. Esto también se especifica en el protocolo RAS.

La colección de Terminales, Gateways, y MCU's manejados por un gatekeeper es llamada una **Zona H.323**, como se ve en la figura 5.4.

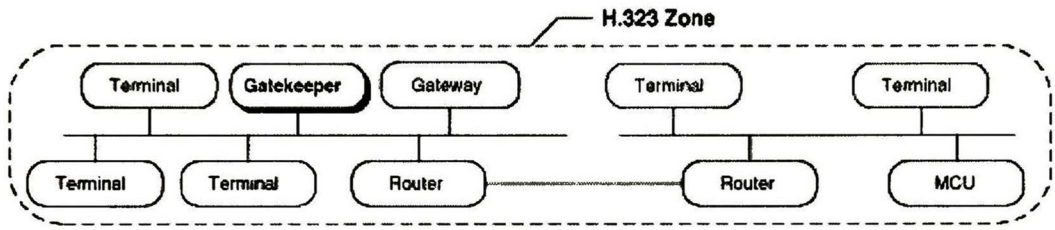


Fig. 5.4

Una facilidad, opcional pero deseable, es el ruteo de las llamadas por medio del gatekeeper. Esta facilidad puede ser utilizada para la tarificación de llamadas.
Un gatekeeper no es obligatorio en un sistema H.323, pero si está presente, es obligatorio hacer uso de sus servicios.
Un gatekeeper es importante en las conexiones multipoint funcionando como un enrutador de los mensajes H.245 entre terminales y el MC (Multipoint Controller).
Las redes que tengan un gateway deben tener también un gatekeeper.

Las funciones requeridas de una gatekeeper son:

- Traducción de direcciones.
- Control de admisión.
- Control del ancho de banda.
- Manejo de la Zona.

Las funciones opcionales de una gatekeeper son:

- Señalización del control de llamadas. (mediante Q.931)
- Autorización de llamadas.
- Manejo de la llamada.

Un gatekeeper puede ser utilizado para las comunicaciones multipunto. Este puede direccionar una llamada a un Controlador Multipunto (MC) sin necesidad de que el mismo gatekeeper haga el procesamiento de las señales de control ni de mezclado de señales.

5.2.2.7 Multipoint Control Unit (MCU)

Permite conferencias entre tres o más terminales. MCU consiste de un Multipoint Controller (MC) y cero o más Multipoint Processors (MP).
MC maneja la negociación de H.245 entre todas las terminales para determinar las capacidades comunes para el procesamiento de audio y video. El MC también controla los recursos de conferencia para determinar cuales, si existen, flujos de audio y video serán multicast.
El MC no trata directamente con los flujos. Esta labor es ejecutada por el MP, el cual mezcla, conmuta y procesa flujos de audio, video y datos.

Si se juntan un MC y un MP se forma un MCU. Y puede localizarse en un dispositivo dedicado o como parte de otra terminal.

5.2.2.8 Conferencias Multipunto

Las capacidades de conferencia multipunto son manejadas de una variedad de maneras y configuraciones en H.323. la recomendación utiliza los conceptos de conferencia centralizada y descentralizada como se describe en la figura 5.5

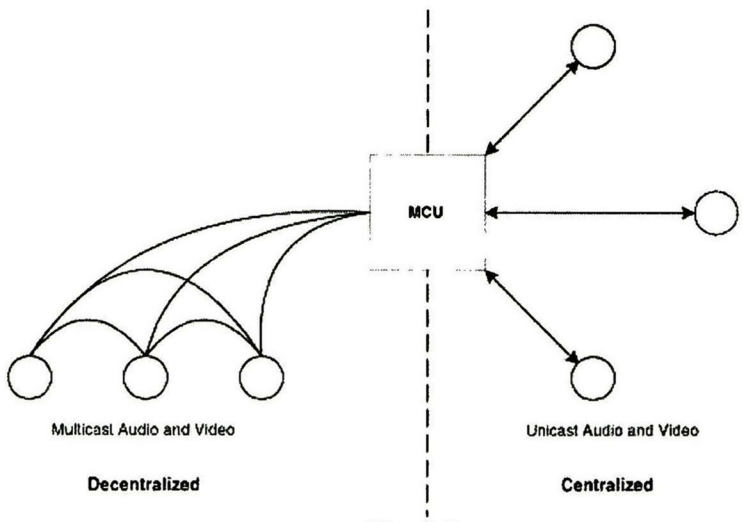


Fig. 5.5
Tipos de Conferencia múltiple

5.2.2.8.1 Conferencia multipunto centralizada.
Para facilitar la conferencia multipunto se requiere la existencia de un MCU. Todas las terminales pueden enviar audio, video, datos y flujos de control al MCU en una forma punto a punto. El MC centralmente maneja la conferencia utilizando las funciones de control de H.245 que también define las capacidades de cada terminal. El MP hace la conmutación y mezclado de flujos y redistribuye los resultados.

5.2.2.8.2 Conferencia multipunto descentralizada.
Esta puede hacer uso de la tecnología multicast. En este tipo de conferencia, las terminales H.323 participantes hacen el multicast por su propia cuenta

Una ventaja de utilizar la conferencia centralizada es que todas las terminales H.323 soportan la comunicación punto a punto. el MCU puede proporcionar varias conexiones unicast a los participantes y no requerir capacidades especiales de red.

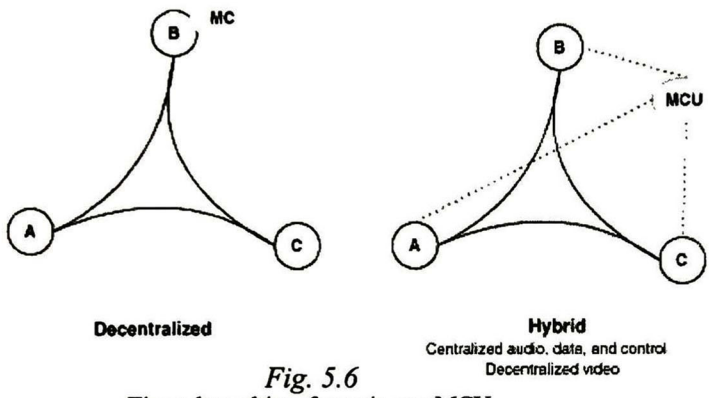


Fig. 5.6
Tipos de multiconferencia con MCU

5.2.2.9 IP Networking y Conferencia Multimedia.

H.323 utiliza tanto comunicaciones confiables como no confiables. Las señales de control y datos requieren de un transporte confiable ya que estas señales deben recibirse en el orden en que fueron enviadas y pueden perderse.

Los flujos de audio y video pierden su valor con el tiempo. Si un paquete se retrasa, puede ya no ser útil para el usuario. Debido a esto, las señales de audio y video utilizan un transporte no confiable como un compromiso entre velocidad de transmisión y procesamiento necesaria para garantizar la confiabilidad de un canal TCP/IP.

Las comunicaciones por canal confiable (TCP) son:

- H.245 Control Channel [39]
- T.120 data channels [43]
- Call Signaling Channel

Las comunicaciones por canal no confiable (UDP) son:
- Audio
- Video
- RAS channel

En conferencias basadas en flujos múltiples, la comunicación no confiable se hace vía UDP utilizando IP-Multicast y RTP. IP-Multicast es un protocolo que utiliza UDP. RTP trabaja sobre IP-Multicast. RTCP es utilizado para controlar RTP.
Es posible utilizar Resource Reservation Protocol (RSVP) para facilitar la llegada de paquetes.

5.2.2.10 Comunicaciones en H.323

Después que la configuración de la llamada está completa, todas las comunicaciones entre dos terminales, toman lugar sobre los canales lógicos que se abren utilizando los procedimientos de H.245. Existe un solo canal lógico para H.245 y es el canal 0. El audio, el video y los datos son enviados por canales lógicos separados y H.323 soporta múltiples canales para cada medio.

Las comunicaciones de H.323 pueden considerarse como una mezcla de audio, video, datos y señales de control. Se requieren las capacidades de audio, configuración de llamada Q.931, protocolo de control RAS y la señalización H.245. Todas las otras capacidades, incluyendo audio y video y datos son opcionales.

Cuando es posible utilizar varios algoritmos, los algoritmos a usar por los codificadores durante una sesión son derivados de la información pasada por el *decoder* durante intercambio de capacidades de H.245. las terminales H.323 son también capaces de operaciones asimétricas (enviar en un formato y recibir en otro) y pueden enviar y recibir más de un canal de información.

5.2.2.11 Control

Es la parte central de las terminales. Entre sus funciones se encuentran:
- Señalización de llamadas.
- Intercambio de características.
- Manejo de canal lógico.
- Empaquetamiento y desempaquetamiento de los flujos de datos.
- Manejo de la red, secuenciación de paquetes, detección y corrección de errores.

Todas las señales de control de audio y video pasan a través de la capa de control que da formato a los flujos de datos dentro de mensajes para salir a la interfaz de red. El proceso inverso toma lugar en los flujos de entrada. Esta capa solamente realiza un encuadrado (framing) lógico, secuenciación, detección y corrección de errores para cada medio.

Estas funciones son implementadas por:
- **H.245** (control Protocol for Multimedia Communications) es un canal confiable que transporta mensajes de control que gobiernan las operaciones de la entidades H.323, incluyendo intercambio de capacidades, apertura y cerrado de canales lógicos, solicitud de preferencias, mensajes de control de flujo y comandos en general. El intercambio de características es una de las capacidades principales. H.245 provee formas de recibir y

transmitir capacidades así como métodos para describir estos detalles a otras terminales H.323. existe un solo canal H.245 por llamada.

- **Q.931** (ISDN user-network interface layer 3) Especificación para control básico de la llamada.
- **RAS Channel** (Registration, Admission, Status) Canal no confiable utilizado para manejo de registros, admisiones, mensajes de estado y cambio de ancho de banda. Sigue la especificación H.225. esta especificación no es utilizada si no existe un gatekeeper en el sistema.
- **RTP (Real Time Protocol)**
- **RTCP(Real Time Control Protocolo)** en una sesión multimedia, el audio y el video son transportados en sesiones RTP separadas con sus propios paquetes RTCP.

5.2.2.12 Audio.

Las terminales H.323 DEBEN dar soporte al estándar G.711 para compresión de voz. Otros estándares son opcionales.

- **G.711** Pulse Code Modulation (PCM) de voz a 56 o 64kbps.
- **G.722** codificación de audio de 7kHz con 64kbps.
- **G.723.1** Dual rate speech coder a 5.3 y 6.3 kbps.
- **G.728** codificación a 16kbps utilizando low-delay code excited linear prediction.
- **G.729** codificación a 8kbps utilizando conjugate structure algebraic-code-excited linear prediction.

5.2.2.13 Video.

Todas las terminales H.323 que soporten video, lo deben hacer utilizando codecs H.261. y el soporte para H.263 es opcional.

5.2.2.14 Datos.

La conferencia de datos es opcional e incluye la transferencia de archivos, pizarrones compartidos, etc. Esta transmisión se hace mediante el protocolo T.120.

Además de los canales lógicos utilizados para los flujos de audio y video, es necesario abrir un canal lógico separado para RTCP. Este canal de control es primeramente utilizado para proveer mecanismos de retroalimentación para la calidad de servicio. La fuente deberá utilizar estos datos para adaptar sus esquemas de codificación o buffering.

Los canales lógicos son multiplexados en la capa de dirección de transporte destino (dirección de red y número de puerto). Por ejemplo, cada canal lógico para H.245, T.120, audio. Video y RTCP es enviado a una dirección destino de transporte distinta.

Las figuras 5.7, 5.8 y 5.9 muestra la pila de protocolos para H.323, cada figura a diferente nivel y perspectiva.

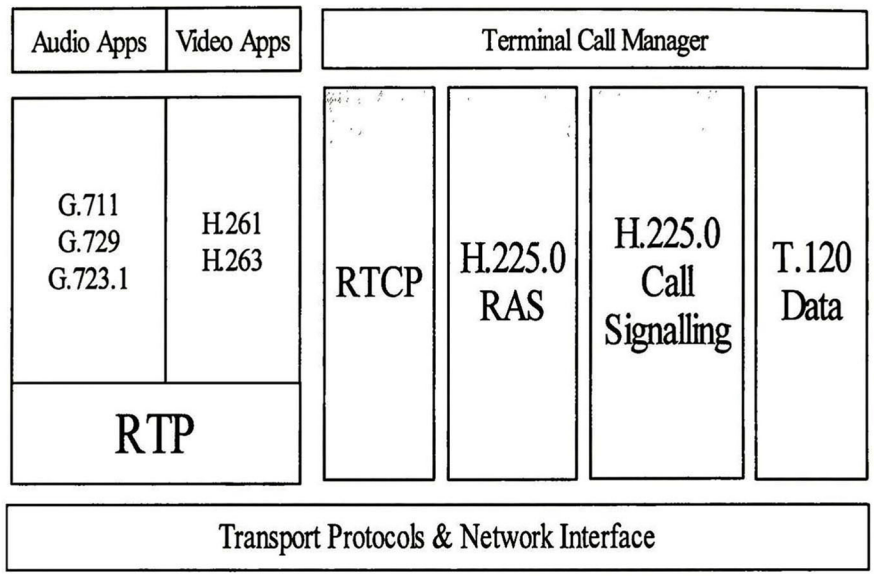


Fig. 5.7

Organización abreviada de los componentes H.323

Audio Apps	Video Apps	Other Media	Terminal Call Manager				Data Apps
G.711 G.722 G.723.0 G.723.1 G.728 G.729	H.261 H.263		RTCP	H.225.0 RAS Channel	H.225.0 Call Signalling	H.245 Control Channel	T.124
RTP					X.224 Class 0		T.125
Unreliable Transport (UDP)					Reliable Transport (TCP)		T.123
Network Layer (IP)							
Link Layer (IEEE 802.3)							
Physical Layer (IEEE 802.3)							

Fig. 5.9

Organización extendida de los componentes H.323

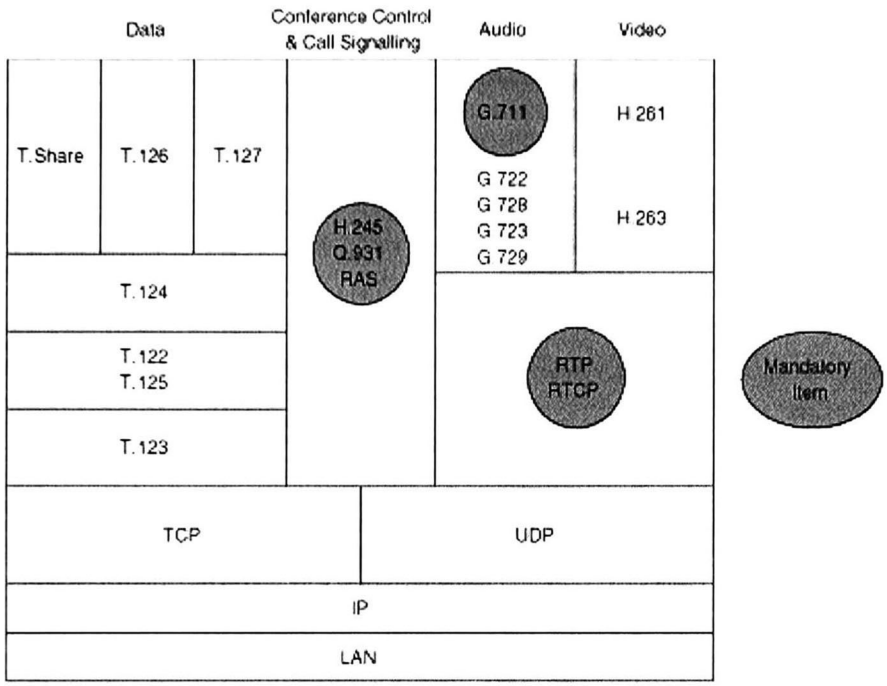


Fig5.8
Organización de los componentes H.323 en módulos

5.2.2.15 Modelo de llamada y configuración de llamada.

H.323 define dos modelos básicos para la señalización de la llamada. El primer modelo es la señalización directa, donde la señalización toma lugar directamente entre las terminales. El segundo modelo es el de señalización de llamada encaminado por un gatekeeper. En el que el gatekeeper controla y libera toda la señalización de la llamada. El modelo encaminado por un gatekeeper fue desarrollado para proporcionar capacidades de conferencia multipunto en las implementaciones en las que las terminales no cuentan con un MC. En estos casos, el gatekeeper puede contener el MC, de esta manera, cualquier llamada entre dos terminales puede ser expandida a una conferencia multipunto [21].

La configuración de la llamada toma lugar en varios pasos. Primero, si la implementación utiliza gatekeepers, la terminal solicitante debe pedir permiso a al GK de establecer una llamada, esto se hace con H.225 [fig. 5.10]. Si el GK permite la llamada, responde con un ACF (Acknowledge Forward), de otra manera, lo rechaza utilizando el mensaje ARJ (Acknowledge Rejected). En caso de que no exista GK o que el permiso es recibido, la terminal solicitante envía un mensaje *setup* a la terminal deseada. La terminal llamada, contesta el mensaje recibido con *call proceeding*, indicando que está procesando el mensaje. Si la terminal solicitada es capaz de recibir la llamada, debe primero pedir permiso al GK (de la misma manera que el solicitante). Después del permiso, la terminal

envía un mensaje *alerting* a la terminal solicitante indicando que el usuario está siendo notificado de la llamada entrante. Si el usuario responde la llamada, la terminal solicitada envía el mensaje *connect* a la solicitante. Ambas terminales establecen el canal H.245 de control, inicia el intercambio de capacidades y abren canales lógicos para la transmisión de los medios [3].
Este procedimiento se muestra en la figura 5.10, mientras que la figura 5.11 muestra un diagrama con el procedimiento completo de una llamada, desde la solicitud hasta la finalización.

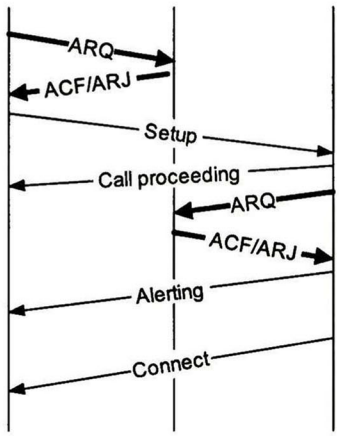


Fig. 5.10
Primitivas de comunicación H.323

5.2.2.16 Calidad de servicio.

H.323 no provee mecanismo alguno para garantizar la calidad de servicio de video o voz en una LAN. De cualquier manera, provee algunas herramientas para el control de la calidad de servicio (sin llegar a garantizarlas). H.323 especifica que el control y los datos deben ser enviados por canales confiables, tales como TCP. Los canales de tiempo real para audio y video utilizan servicios de transporte de tipo *mejor esfuerzo*, tales como UDP que no garantizan la entrega de paquetes y no implementan confirmación y retransmisión. El Jitter, la pérdida de paquetes y los congestionamientos de la red pueden causar efectos adversos a estos canales no confiables [3].

H.245 y RTCP proveen mecanismos opcionales para solicitar la retransmisión de paquetes.

El gatekeeper provee mecanismos para la administración de la red, número de usuarios de H.323 y manejo de ancho de banda. Aunque esto no garantiza el uso de ancho de banda de una llamada, si controla el impacto de las llamadas H.323 en el tráfico de una LAN.

En aquellas redes que contienen enrutadores y que soportan protocolos de reservación, las terminales H.323pueden hacer uso de estos servicios antes o durante la configuración de una llamada. Esto reservará recursos de red a lo largo de la ruta de las terminales que están llamando. Un protocolo de este tipo es RSVP o Resource Reservation Protocol [3].

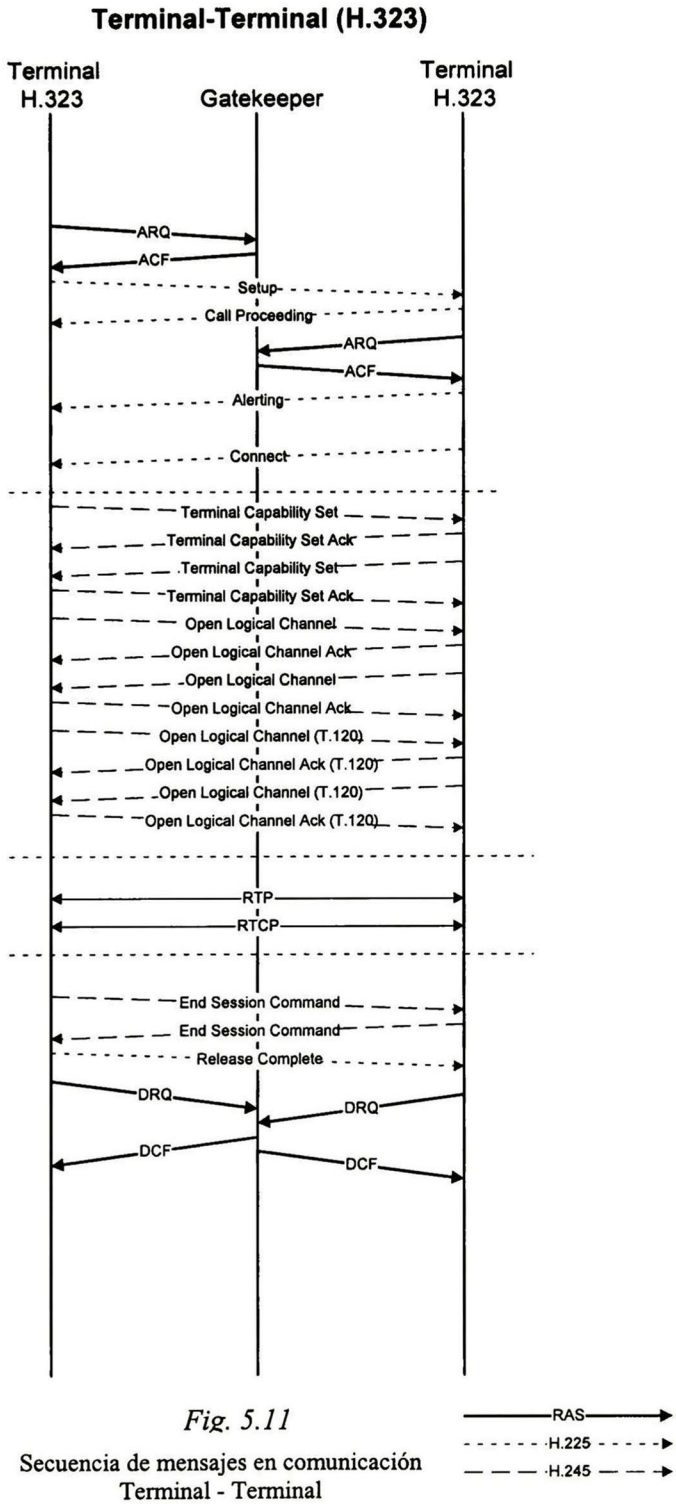


Fig. 5.11

Secuencia de mensajes en comunicación Terminal - Terminal

6

Metodología Formal de Análisis y Diseño LAPEM LOTOS Assisted Protocol Engineering Method

6.1 LOTOS

6.1.1 Introducción

El diseño de sistemas, incluyendo la implementación y desarrollo de software, debería estar basado en métodos formales con la finalidad de alcanzar un diseño correcto.

El diseño de sistemas distribuidos envuelve el diseño arquitectónico y las fases de implementación subsecuentes. Enfatizamos en el uso de modelos formales para el desarrollo de sistemas para obtener soluciones independientes de la implementación o implementaciones intermedias [34].

6.1.2 Cálculo para Sistemas de Comunicación CCS

La base de LOTOS está en entes anónimos y síncronos que envían y reciben valores sobre puertos de comunicación (gates). Estos entes se ensamblan por medio de estas compuertas de comunicación.

Las compuertas de comunicación son canales limitados en visión, es decir, no es posible saber nada más de la entidad que nos envía información, que los mensajes que nos son enviados.

CCS es la parte matemática de LOTOS, es un lenguaje de cálculo de predicados que tiene las siguientes características:

- El agente transmisor está bloqueado hasta la recepción de su mensaje por el receptor.
- El objeto transmisor está bloqueado hasta que su requisición es procesada por el objeto llamado y hasta que el resultado es regresado.
- Los procesos y dispositivos son modelados de manera similar.
- No distingue entre agentes y redes de agentes. Un agente puede ser monolítico o compuesto.
- Se auxilia de máquinas de estados finitos para su validación.
- Se puede utilizar el cálculo de equivalencias y la bisimulación para la validación

6.1.2.1 Algebra de procesos

En este contexto, es un lenguaje para describir los componentes de una red en LOTOS.
Es auxiliada por EFSM para el modelado de comportamientos.
Mediante este lenguaje nos es posible agregar ciertos comportamientos a los módulos, como son:

- Restricciones
- Composición paralela
- Recursividad

6.1.3 Modelado Abstracto de Sistemas y de su comportamieno.

LOTOS(Language of Temporary Order Specification) es un lenguaje formal que permite modelar sistemas considerándolos como una caja negra con un cierto número de *"compuertas"* que pueden ser vistas desde el *"ambiente"*
El primer paso cuando se desea especificar un sistema es la selección de los aspectos relevantes y la elección de los mismos que serán mapeados en compuertas. Los eventos que son abstraídos para describir un sistema definen la granularidad del comportamiento del sistema. Por ejemplo, para el modelo de un semáforo pueden imaginarse tres compuertas R, Y,G (Red ,Yellow Green) ,una para cada luz de la señal. Esta elección ha sido hecha desde el punto de vista de un conductor típico. Es notorio que se han abstraído una gran cantidad de aspectos, tales como el mecanismo de temporización, la fuente de energía, etc.
El sistema denota su comportamiento por la activación de sus compuertas. Un evento o acción observable corresponde a la activación de una compuerta. El comportamiento del sistema es especificado describiendo todas las posibles secuencias de eventos que el sistema puede ofrecer al ambiente. Usando el ejemplo del semáforo anterior su comportamiento puede ser especificado por la siguiente secuencia de eventos:

G Y R G Y R G Y R G Y R . .

Los eventos siempre ocurren secuencialmente. En otras palabras, lo eventos nunca ocurren simultáneamente.

6.1.3.1 Input/Output(E/S)

- Una instancia de una entrada o salida es modelada por medio de un *"evento"* o *"acción"*
- En cualquier caso la entrada/salida de datos toma lugar a través de las compuertas.
- Un sistema cualquiera tiene una interface con un número fijo de compuertas.
- En LOTOS, las compuertas se nombran con identificadores tipo Pascal :
- Dato_in , BUScpu.

6.1.3.2 Entrada

La entrada es modelada por medio de la *aceptación de un valor* desde el ambiente hacia una compuerta. El valor recibido es guardado en una *variable*, esta variable es declarada localmente en el punto de aceptación del valor.. La declaración tiene un *identificador de variable* y un *sort*.

- Sintaxis de de una entrada genérica (aceptación de un valor) :

```
<nombre_compuerta> ? <variable_id> : <nombre_sort>
```

- ejemplo de una entrada :

```
keyboard_in ? tecla : nat
```

6.1.3.3 Salida

La salida de un sistema es modelada *ofreciendo un valor* al ambiente a través de una compuerta, el valor ofrecido es una *expresión*. Una expresión está formada de *operadores* y variables.

- Sintaxis de una salida genérica(ofrecimiento de un valor) :
  `<nombre_compuerta> ! <expresión>`
- Ejemplo de una salida :
  `Keyboard_out ! ( base * altura )`

6.1.3.4 Prefijo de acción(;)

El prefijo de acción denota secuencialidad, además compone un evento y la descripción de su comportamiento.

- Ejemplo :

`accion ; B` Significa que el sistema ejecuta la acción y entonces se comporta como B.

La secuencialidad entre eventos es denotada por el operador ; , llamado *prefijo de acción.* Este operador compone una acción *a* con una expresión de comportamiento *B* El significado intuitivo es que el sistema aceptará inicialmente el evento *a* comportándose en lo sucesivo como *B*. Es importante recordar que este operador prefijo de acción no toma dos comportamientos como argumento, como la mayoría de los operadores. Sus argumentos son la denotación de un evento y una expresión de comportamiento. Los eventos son usualmente representados como una línea etiquetada con el nombre del evento. Las expresiones de comportamiento son usualmente representadas como un triángulo (representación secuencial) o como una caja (composición paralela).
Ejemplos del uso del prefijo de acción.

1. Un ruteador IP recibe un datagramas y lo debe rutear ;
2. El ruteador IP se ve forzado a segmentar el datagrama;

Este comportamiento puede ser abstraído así:

```
  Net1_in ? datagrama : ip_dtgrm
; Net2_out ! Primer_Segmento(datagrama)
; Net2_out ! Segundo_Segmento(datagrama)
; Net2_out ! Tercer_segmento(datagrama)
;
```

6.1.3.5 Choice "[ ]"

El operador choice realiza la composición de dos expresiones de comportamiento C1 , C2 para formar otra que se puede comportar como cualquiera de ellas. La selección entre C1 y C2 depende del primer evento que ocurra. Una vez que un evento(E/S generalmente) perteneciente a C1 o C2 ocurre, el comportamiento de la expresión será el de la expresión a la cual pertenece el evento. El otro comportamiento es desechado.
Notemos que usando el prefijo de acción y el operador de choice es posible describir el comportamiento de un sistema como un árbol de eventos. El operador ";" hace al árbol más profundo (con más niveles) y el operador "[ ]" permite la adición de ramas a los nodos de los árboles.

Los operadores ";" y "[ ]" son los operadores básicos de LOTOS. La semántica de todos los demás operadores puede ser representada en términos de un árbol equivalente usando sólo estos dos operadores.

<Comp1> [] <Comp2> significa que el sistema se comporta como alguna de las dos alternativas.

- Ejemplo de choice :

Un ruteador IP puede recibir datagramas provenientes de la Net 1 o de la Net 2 .

```
( Net1_in ? datagrama    ip_dtgrm
; Net2_out   ! Primer_Segmento(datagrama)
;         )
[]
( Net2_in ? datagrama : ip_dtgrm
; Net1_out   ! Primer_Segmento(datagrama)
;         )
```

Inicialmente los eventos Net1_in ... y Net2_in ... son ofrecidas al ambiente, una vez que el ambiente selecciona uno, solamente se ofrece el siguiente evento secuencial.

6.1.3.6 Guarda : "[<exp1>= <exp2>]->"

Una guarda es un predicado sobre valores que sirven de prefijo a un comportamiento.
Su uso típico es el de seleccionar internamente entre acciones en un proceso de choice.
Ejemplo de una máquina que despacha monedas :

```
( [monedero_lleno = true] ->
;  Monedero ! dar_dinero
;           )
[]
( [monedero_lleno    false] ->
;  Pantalla_Monedero ! mensaje_no_hay_dinero
;           )
```

6.1.3.7 Procesos

El significado intuitivo de la abstracción de procesos es muy similar al concepto de procedimientos (más precisamente corrutinas) en los lenguajes de programación imperativa usuales. Es posible definir un comportamiento asignarle un nombre y después invocar múltiples instancias de él.

Una instanciación de un proceso es una instancia en ejecución de una *"definición de proceso"*
Una definición de proceso es la descripción del comportamiento de un subsistema.
La definición de procesos es la única manera en LOTOS de especificar comportamientos recursivos o iterativos(no hay sntencias tales como *while , for goto*). En la definición de procesos es necesario declarar las compuertas formales a través de las cuales se interactuará con el sistema. Cuando el proceso es instanciado, es posible sustituir la lista de compuertas formales por la lista de compuertas actual, este mecanismo es llamado *"renombrado de compuertas"* y es muy similar al concepto de parámetros formales y parámetros actuales en los lenguajes de programación convencionales.

- La instanciación de un proceso es la ejecución de una instancia de una "*definición de proceso*".
- Una definición de proceso es la descripción del comportamiento de un subsistema.
- La construcción de procesos en LOTOS sirve para tres propósitos principales :

i. Representación de la recursividad (ciclos y comportamiento infinito).
ii. Crear abstracciones de comportamiento y jerarquías(top-down).
iii. Renombrado de compuertas.

6.1.3.8 Definición de procesos e Instanciación

- Definición de procesos :

```
PROCESS
<nombre_proc> [<compuertas formales>] (<parámetros>) : <funct> :=
      <comportamiento>
WHERE
      <definiciones locales>
ENDPROC
```

- Instanciación de procesos

```
<nombre_proc> [<compuertas actuales>] (<valores de los parámetros>)
```

- Ejemplo (recursividad):

```
PROCESS
IP_ROUTER [Net1_in,Net1_out,Net2_in,Net2_out]   NOEXIT :=
( Net1_in ? datagrama   ip_dtgrm
; Net2_out ! datagrama
; IP_ROUTER [Net1_in,Net2_out,Net2_in,Net2_out]
)
[]
( Net2_in ? datagrama   ip_dtgrm
; Net1_out ! datagrama
; IP_ROUTER [Net1_in,Net2_out,Net2_in,Net2_out]
)
ENDPROC
```

6.1.4 Introducción a los tipos de datos

tipos : constructores para la encapsulación de declaraciones y definiciones.
sorts : conjuntos disjuntos de valores.
operaciones : declaraciones de funciones(las constantes son un tipo particular).
ecuaciones : semántica de las operaciones (puramente funcional, en otras palabras sin memoria).

6.1.4.1 Librería de tipos de datos

Contiene los tipos de datos y construcciones más usuales que se usan dentro de una especificación en LOTOS.

ejemplo de un tipo :

```
TYPE Boolean IS
    SORTS
      bool
OPNS
     true   false                                          -> bool
     not                                                   -> bool
     _and_    _or_    _xor _   _iff _    :   bool   bool -> bool
     _equal_   _ne _                     :   bool   bool -> bool
EQNS
    .
ENDTYPE
```

6.1.4.2 Estructura de una especificación

```
SPECIFICATION
<nombre> [<compuertas>] (<params>) : <funcionalidad>
    <tipos de datos>
BEHAVIOUR
    <comportamiento>
WHERE
    <definición de tipos locales>
ENDSPEC
```

6.1.5 Máquinas de Estado Finito Extendidas (Extended Finite State Machines o EFSM)

Se denominan así un tipo especial de máquinas de estado finito con variables auxiliares para datos. Este tipo especial de autómatas son usados frecuentemente para especificar e implementar protocolos.

Este tipo de máquinas es definido por :

i) un conjunto de *estados*.
ii) Un conjunto de *variables de estado*.
iii) Un conjunto de *entradas* para el autómata.
iv) Un conjunto de *salidas* para el autómata.
v) Un conjunto de transiciones extendidas.

Una transición extendida es definida por un estado inicial, la entrada que dispara la transición, un predicado que la habilita, la salida, un conjunto de acciones y el nuevo estado.

6.1.5.1 EFSM en LOTOS

Una definición de proceso es realizada para cada estado.
Las variables de cada estado se mapean a parámetros en cada proceso.
Las transiciones desde un estado se mapean a una elección de eventos seguidas de una instanciación de proceso.
Cada entrada que dispara una transición es mapeada a un evento.
Los predicados sobre variables que deben retener el disparo de una transción son mapeados a una guarda precediendo el evento.
La salida asociada a una transición es mapeada a un evento secuencial después del disparo del evento.
El nuevo estado se mapea a la instanciación del proceso correspondiente.

La actualización de las variables se mapean a los parámetros de los procesos instanciados.

6.1.5.2 Plantilla de una EFSM en LOTOS

```
Process EstadoInicial [<compuertas>]  (* Estado Inicial *)
        (var1 : T1, ...   varn : Tn ) (* variables de contexto*)
:=
( [Predicado]->                    (* Predicado de habilitación*)
  <events> ;                       (* Interacción              *)
  Edo_Siguiente[<compuertas>]   (*Instanciación del estado sig.*)
      (Expr1,...,ExprN)         (* variables actualizadas      *)
[]                              (* Siguiente transición       *)
```

6.1.5.3 Interleaving "|||"

El entrelazado es un operador *paralelo*, en LOTOS los operadores paralelos modelan la *concurrencia*. El entrelazado representa la composición concurrente sin interacción; dentro de LOTOS se considera que no existe concurrencia verdadera sólo un tipo especial *concurrencia entrelazada*.

En otras palabras : Los dos subsistemas concurrentes *entrelazan* sus eventos.

Ejemplo de entrelazado :

```
Process IP_ROUTER [Net1_in,Net1_out,Net2_in,Net2_out]: noexit :=
  FORWARD[Net1_in,Net2_out]
|||
  FORWARD[Net2_in,Net1_out]
where
process FORWARD[Net_in,Net_out] : noexit :=
   Net_in  ? datagrama : ip_dtgrm
  ;Net_out ? datagrama
  ;FORWARD[Net_in,Net_out]
  endproc
endproc
```

6.2 LAPEM

6.2.1 Introducción

El ciclo de vida de un sistema distribuido (análisis, diseño e implementación) debería estar basado en métodos formales para obtener mejores resultados [34].

Este capítulo describe un método basado en el lenguaje de especificación LOTOS para el proceso de análisis y diseño de un sistema distribuido, en especial los protocolos de comunicaciones.

La parte central de este capítulo es la etapa de análisis (comprensión del problema) y diseño (solución del problema) de un sistema de comunicaciones en un sistema distribuido. Está basado en máquinas de estados finitos y cronogramas para modelar comportamientos y

secuencia ordenada de mensajes respectivamente. Mostramos también la forma de convertir estas máquinas de estados finitos al lenguaje LOTOS.

Hemos elegido el lenguaje LOTOS por su capacidad de modelar procesos formalmente desde el punto de vista de un usuario externo, esto es llamado *blackboxes*. LOTOS permite también el uso de una forma de trabajo *top-down*. Otra característica fuerte de LOTOS es la representación de comportamientos en paralelo.

La existencia de varios simuladores *freeware* y comerciales es otra de las ventajas de este lenguaje.

La principal desventaja de LOTOS es que no existe una metodología estándar de resolver un problema orientando la solución a este lenguaje específico. En [33], [34], [35], [36], [26], y [27] se muestran algunas aproximaciones de formas de trabajo para LOTOS, algunas de ellas utilizan máquinas de estados finitos y otras utilizan una metodología clásica del modelo de cascada de ingeniería de software.

Estas aproximaciones no son suficientes para este trabajo ya que aquí consideramos procesos independientes que son ejecutados sobre máquinas independientes y que además pueden tener problemas en la comunicación o la forma de esta puede cambiar. Debido a esto, decidimos crear una nueva metodología de análisis y diseño de sistemas distribuidos y protocolos de comunicaciones orientando la solución al lenguaje LOTOS. Llamamos a esta metodología LAPEM (LOTOS Assisted Protocol Engineering Method).

6.2.2 La metodología

La metodología de diseño de sistemas en LOTOS es normalmente dada por un refinamiento iterativo y progresivo de los bloques (black boxes) especificándolos en cada iteración más a detalle (convertirlos en white boxes), todo esto, ayudado por el lenguaje, que expresa capacidades y propiedades de los elementos que se relacionan. Esto se muestra en la gráfica siguiente:

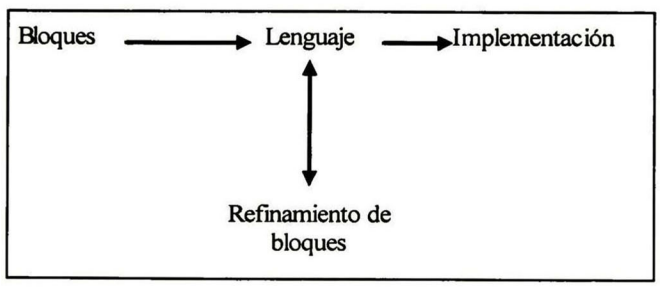


Fig. 6.1
Metodología tradicional de análisis y diseño

Una metodología así trae varios problemas, ya que no está basada en una especificación ni tiene una diferencia clara entre análisis y diseño del sistema, en cambio, mezcla análisis y diseño en un solo proceso que no tiene un final claro. En otras palabras, es posible iterar de manera redundante ante una solución ya encontrada o, en caso contrario, crear módulos demasiado grandes.

LAPEM es una metodología de análisis y diseño orientada al comportamiento de procesos comunicantes. Es el resultado de una mezcla de elementos ESTELLE y FUSION en un orden definido por nosotros. Con esta metodología, definimos el modelo global del sistema con diagramas de bloques (iguales a los de ESTELLE) y los refinamos de manera progresiva sin tomar en cuenta el lenguaje ESTELLE o la plataforma de implementación.

A continuación, una vez definidos todos los módulos que interactuarán y decidiendo que una mayor división no traería beneficios al sistema, encontramos los mensajes que intercambian estos módulos.

Pasamos enseguida a mostrar los mensajes mediante escenarios o cronogramas (extraídos de FUSION), los que nos muestran la secuencia temporal de estos mensajes. Después, expresamos el comportamiento de estos módulos mediante Máquinas de Estados Finitos Extendidas (EFSM por sus siglas en inglés) y traducimos cada uno de los autómatas al lenguaje LOTOS. Finalmente llegamos a la fase de implementación del sistema.

Ordenando esta secuencia de acciones, podemos decir que la metodología consiste de 7 pasos, algunos de ellos iterativos.

1. **Diagrama de bloques inicial.** Partiendo de una especificación, obtenemos un diagrama de bloques que modela los principales componentes del sistema así como los elementos externos con los que se comunica
2. **Refinamiento abstracto.** A diferencia de las formas de análisis y diseño en LOTOS tradicionales, nosotros hacemos un refinamiento basándonos solamente en gráficos, evitando así el uso innecesario, en esta etapa, del lenguaje LOTOS.
   El criterio de división de los módulos es el siguiente:
   - Se debe crear un módulo siempre que se necesite un nivel diferente de abstracción
   - Cada módulo debe realizar una función bien definida
   - Los límites de los módulos deben elegirse a modo de minimizar el flujo de información a través de las interfaces o canales de comunicación
   - La cantidad de módulos debe ser suficiente para no tener que agrupar funciones distintas en una misma capa. Cada módulo debe tener acciones específicas y relacionadas entre sí.
3. **Obtención de mensajes.** Una vez obtenido un diagrama de bloque que cumpla con los criterios anteriores, podemos obtener un listado de los mensajes que pasan en todos canales de comunicación
4. **Cronogramas.** Con el listado anterior, organizamos los mensajes por eventos. Por evento queremos decir, un mensaje que llega desde alguna de las entidades externas y que cambia el estado del sistema o que desencadena algún intercambio de mensajes entre los elementos internos de nuestro sistema, o que da una salida a alguno de los elementos externos.

Esto nos permite darnos cuenta de la falta de módulos o mensajes y poder organizar de una mejor manera las simulaciones posteriores.

5. **Autómatas.** Una vez obtenidos el diagrama de módulos y los cronogramas, modelamos el comportamiento de cada uno de los módulos mediante máquinas de estados finitos. Si se cree necesario, es posible hacer un modelo de máquina de estados finitos extendida que cumpla con los requerimientos del problema. Un ejemplo de esto lo podemos ver en [14].
6. **Lenguaje LOTOS.** La traducción de las máquinas de estado finito a lenguaje LOTOS se puede ver en el Apéndice I.
   En este momento podemos verificar si nuestro diseño cumple con la especificación. Esto mediante la simulación del código LOTOS generado.
7. **Implementación.** Una vez obtenido el código LOTOS que modela al sistema, es posible traducir este código a un lenguaje de programación, por ejemplo C. Hay varios traductores automáticos de código LOTOS a lenguaje C y están disponibles a través del WEB.

Esta secuencia de acciones se muestra en el siguiente esquema:

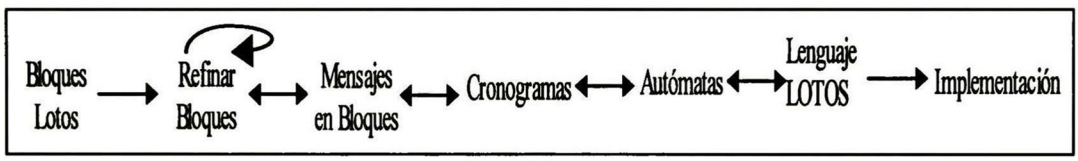


Fig. 6.2
Secuencia de eventos para LAPEM

Este esquema de análisis y diseño nos da la ventaja de poder pensar en el comportamiento del sistema y los elementos que lo componen, sin tener las restricciones que un lenguaje de especificación formal nos impone a este nivel de análisis y diseño.

6.2.3 Un ejemplo

En el siguiente ejemplo podremos ver de manera práctica la metodología anteriormente descrita.
Por cuestiones de espacio, pondremos solo parte de la solución en este capítulo. El ejemplo completo se encuentra en el apéndice I.

6.2.3.1 Definición del problema

Problema: Se desea especificar, analizar y diseñar un protocolo de comunicaciones como FTP, pero simplificándolo para fines demostrativos.
Las características de este protocolo, que llamaremos SFTP (Simplified FTP), se muestran a continuación:

1. El usuario, mediante el *Cliente*, se conecta con el *Servidor*.
2. Para ingresar al servidor, se requiere de un *ID* y un *Password*.
3. El usuario puede ejecutar comandos locales.
4. El usuario puede ejecutar comandos remotos.

5. El usuario puede ejecutar comandos de transferencia, esto es, puede hacer una transferencia de archivos desde el servidor al local.
6. El usuario puede salir.
7. En caso de iniciar una conexión y que el servidor rechace tal solicitud, el usuario deberá iniciar en el paso 1.
8. La ejecución de cualquiera de los comandos por el usuario, generará un aviso o notificación del resultado de la ejecución.

6.2.3.2 Diagrama de módulos

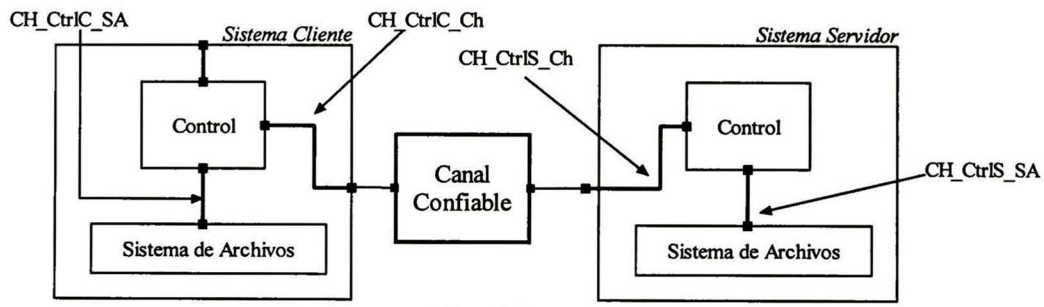


Fig. 6.3
Diagrama de módulos

6.2.3.3 Mensajes entre módulos

Control Cliente → Sistema de Archivos.

- Comando local

Sistema de archivos → Control Cliente.

- Notificación de comando local

Control cliente → Canal confiable.

- Requisición de conexión(dir)
- User(ID)
- Password(passwd)
- Comando transferencia
- Comando remoto
- Requisición de desconexión

Canal confiable → Control Cliente.

- Fin transferencia
- Acceso(si)
- Acceso(no)
- Confirmación comando remoto
- Confirmación comando transferencia
- Confirmación de conexión

- Data(buffer)

6.2.3.4 Escenarios o cronogramas

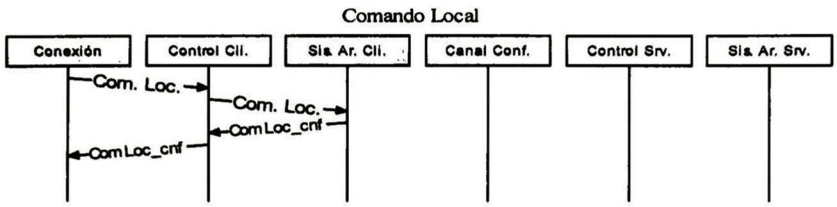


Fig. 6.4

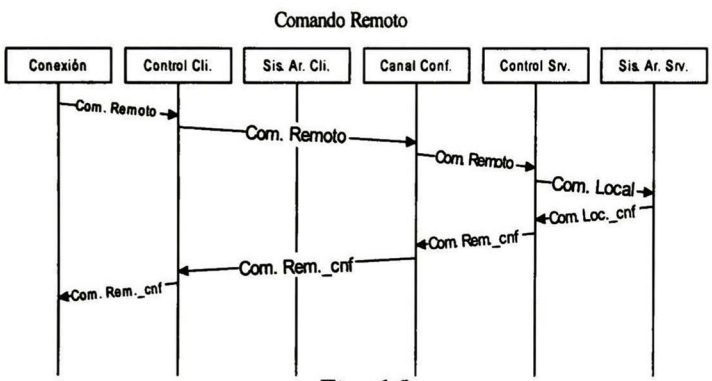
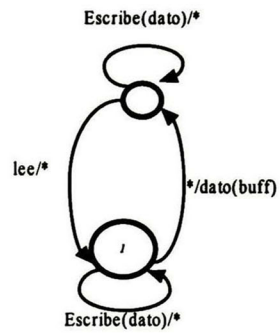


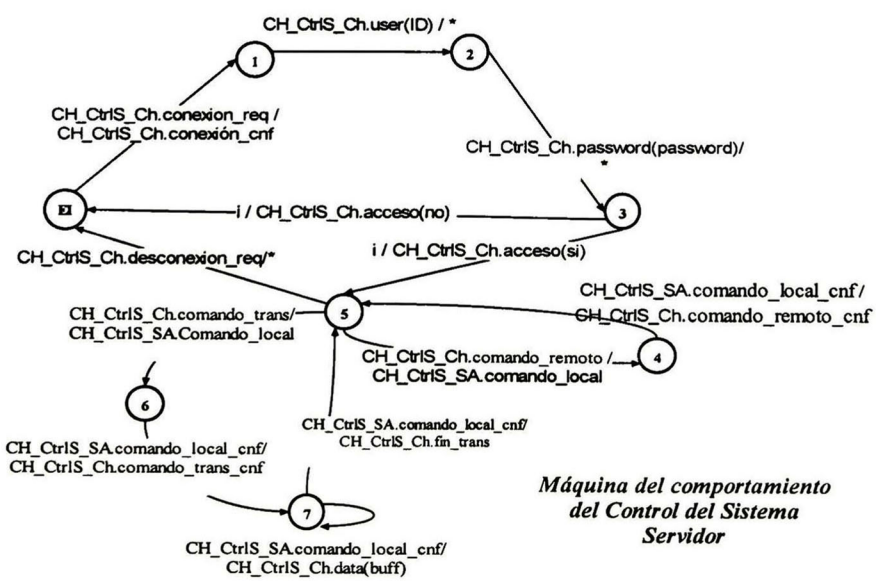
Fig. 6.5



Máquina del comportamiento del Buffer

Fig. 6.7

6.2.3.5 Autómatas



Máquina del comportamiento del Control del Sistema Servidor

Fig. 6.6

6.2.3.6 Código LOTOS

```
SPECIFICATION    Ejemplo    [CH_Usr_Ctrl,    CH_CtrlC_Ch,    CH_CtrlC_Buff,
CH_CtrlC_SA, CH_CtrlS_SA, CH_CtrlS_Ch]: NOEXIT


BEHAVIOR

((((Estado_Inicial_Control_Cte[CH_Usr_Ctrl,  CH_CtrlC_Ch,  CH_CtrlC_Buff,
CH_CtrlC_SA](0)  |[CH_Usr_Ctrl]|

      Estado_Inicial_Usuario[CH_Usr_Ctrl])  |[CH_CtrlC_Buff]|
      Estado_Inicial_Buffer[CH_CtrlC_Buff](0)) |[CH_CtrlC_SA]|
      Estado_Inicial_Sistema_ArchC[CH_CtrlC_SA](0)) |[CH_CtrlC_Ch]|
      Estado_Inicial_Canal_Conf[CH_CtrlC_Ch, CH_CtrlS_Ch])

      |[CH_CtrlS_Ch]|

      (Estado_Inicial_Control_Serv[CH_CtrlS_Ch,           CH_CtrlS_SA](0)
|[CH_CtrlS_SA]|
      Estado_Inicial_Sistema_ArchS[CH_CtrlS_SA](0))


WHERE
```

6.2.3.7 Simulación

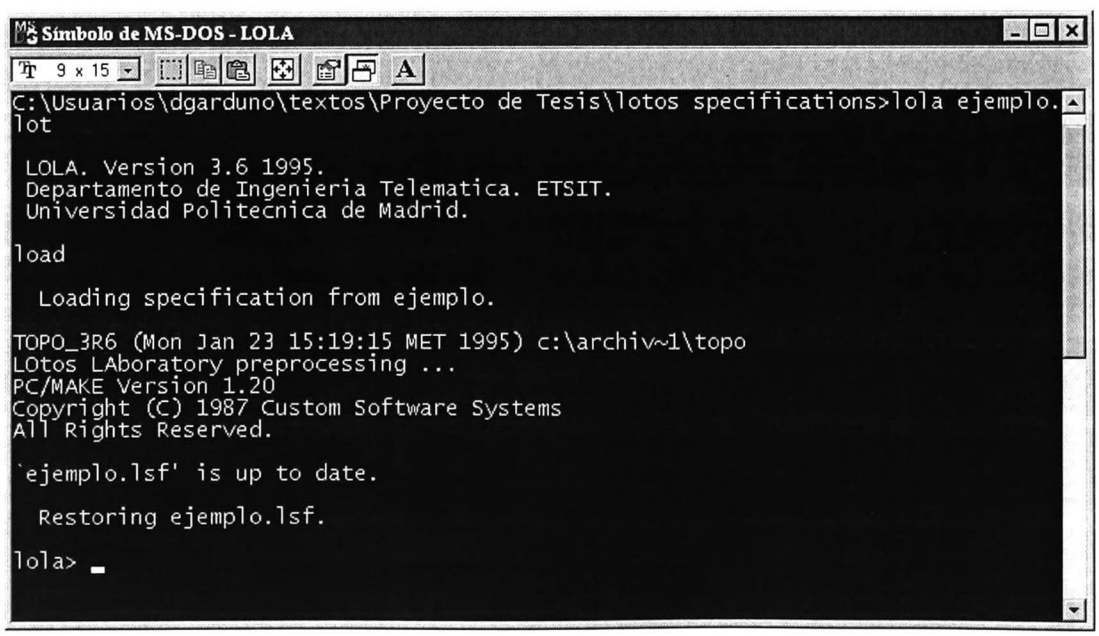


Fig. 6.7

Ventana de compilación exitosa

7

Arquitectura de Software Propuesta

7.1 Algoritmo de sincronización

7.1.1 Introducción

Los objetos de datos envueltos en las presentaciones multimedia pueden tener requerimientos temporales de reproducción estrictos. Cada uno de estos objetos puede tener relaciones específicas con otros objetos en la misma presentación. Por ejemplo, una presentación de tipo *"slide show"* requiere de componentes auditivos y visuales en un orden específico, además de un puntero. Si estas relaciones temporales son violadas, el resultado puede ser no aceptable para el usuario final.

Dadas unas relaciones temporales para la reproducción de una colección de objetos multimedia, la secuenciación y temporización de la reproducción puede ser alcanzada mediante el uso de un sistema operativo de tiempo real. [5].

El sistema se complica si los datos son transmitidos por una red de computadoras. Esta dificultad es debida a la alta tasa de datos que contiene una presentación multimedia y a la naturaleza compartida de las redes de computadoras.

En este contexto, es necesario escoger un algoritmo que pueda trabajar sobre redes no confiables y manejar datos multimedia de diferentes tipos (continuos y no continuos) y que sea independiente del contenido de información.

Existen otros algoritmos para la sincronización de sistemas multimedia en redes de cómputo. El lector interesado puede consultar [4], [5], [6], [7], [8], [9], [11], [13], [19], [20].

7.1.2 Breve descripción

Dentro de la norma H.323, la sincronización es uno de los elementos de base. De cualquier manera, es uno de los elementos dejados al implementador. Esta es una decisión acertada ya que abre las puertas a la creatividad. Aún cuando existen algoritmos de sincronización en la literatura, algunos de ellos se basan en el manejo de tamaño de buffer, otros en scheduling y otros en estimación del comportamiento de la red.
Hemos preferido hacer un algoritmo propio que combine algunas de las características anteriores. Aunque en este trabajo no se describe a detalle este algoritmo, puede consultar [48].

Este algoritmo es independiente del contenido de los objetos multimedia y de sus características (ej. Frecuencia, tamaño, formato, tipo) y está diseñado para utilizarse como un módulo intermediario para la transmisión de información multimedia en redes no confiables.

El hecho de que no trata el manejo de canales (apertura y cerrado) lo hace más adaptable a nuestro trabajo.

Este algoritmo cubre dos etapas de sincronización, Interflujo e Intraflujo. La primera la resuelve mediante el etiquetado de paquetes de red en la transmisión y la discriminación de paquetes retrasados en la recepción. La sincronización Intraflujo la resuelve mediante la estimación del tiempo interarribo de paquetes y el cambio de tamaño de paquetes de red en la transmisión.

Un termino que se utiliza a lo largo de este capítulo es el factor λ. Este representa el espacio temporal que existe entre dos ULDs contiguos del mismo objeto multimedia en la producción.

7.1.3 Características generales

A continuación se enumeran las características generales de este algoritmo.

1. El algoritmo no crea canales por sí mismo.
2. El algoritmo no destruye canales por sí mismo.
3. La ejecución inicia asumiendo que la estructura de comunicación ya está construida, es decir, los flujos ya están abiertos, los canales reservados y los elementos correspondientes listos para comunicación.
4. El algoritmo no maneja el inicio o fin de sesión. Esta tarea es realizada por algún protocolo de negociación de mayor nivel (en nuestro caso, H.323).
5. El algoritmo no trata con aspectos de negociación de canales (cambios de canales o de características de los mismos).

6. Las características de procesamiento del medio son irrelevantes para el algoritmo (compresión, resolución, muestreo. Frecuencia, etc.).
7. La Calidad de Servicio admisible para cada flujo, es negociada antes de la creación del módulo de sincronización.
8. Si la CdS llega a una cota inferior, previamente establecida, le corresponde al Control negociar un nuevo formato de representación de la información y establecerlo en el procesamiento.
   Este cambio de representación es irrelevante para el algoritmo de sincronización, ya que solo es afectado por el tiempo de generación y no por el formato de la información.
   En nuestro caso, no está presente un nuevo formato, lo que implica que la comunicación no es viable y por tanto, termina la comunicación.
9. Todos los paquetes de un mismo flujo serán etiquetados con un identificador distinto y no decreciente, esto es:
   Sea ID_i el identificador de $Paquete_i$, y
   Sea ID_j el identificador de $Paquete_j$.
   Si $t(Paquete_i) < t(Paquete_j)$ entonces $ID_i < ID_j$ y
   Si $t(Paquete_i) \geq t(Paquete_j)$ entonces $ID_i \geq ID_j$
   **La función t se describe en [48] y regresa el tiempo en que fue generado un LDU.*

10. Existirá una estructura tipo FIFO en el receptor para evitar, en lo posible, que se puedan desechar ULDs y ayudar así a la sincronización.
11. En cuanto llegue un LDU a la FIFO de la recepción, se ordenará con los ULDs existentes en tal estructura.
12. El flujo que tiene λ menor, tiene más posibilidades de llegar, ya que tiene una mayor frecuencia de envío por la red. Debido a esto, es el indicado para tener las etiquetas de los demás, es decir, es quien lleva la pauta para la sincronización Interflujo del resto.
13. Sean a_i el ID del flujo de λ menor, y b_j el ID del flujo de λ mayor. Como se muestra en la figura 7.1.

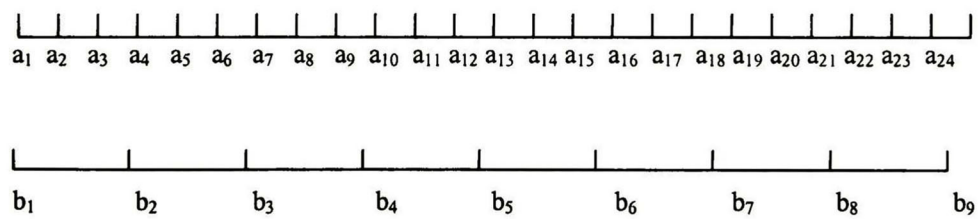


Fig. 7.1
Secuencia de frames etiquetados

Se define un nuevo etiquetado para los paquetes correspondientes a cada flujo:

- El flujo con λ más pequeño, cambiará su etiqueta $\mathbf{a_i}$ por $\mathbf{a_i b_j}$ donde **bj** el la **b** inmediata anterior en tiempo.
- El flujo con λ mayor, quedará como $\mathbf{b_k}$

Así, en la figura anterior el flujo **a** se etiquetaría de la siguiente manera:

$\mathbf{a_1b_1}$, $\mathbf{a_2b_1}$, $\mathbf{a_3b_1}$, $\mathbf{a_4b_2}$, $\mathbf{a_5b_2}$, $\mathbf{a_6b_2}$, $\mathbf{a_7b_3}$, $\mathbf{a_8b_3}$, $\mathbf{a_9b_3}$, $\mathbf{a_{10}b_4}$, $\mathbf{a_{11}b_4}$, $\mathbf{a_{12}b_4}$, $\mathbf{a_{13}b_5}$, etc.

14. Al momento de ser reproducido, el flujo con etiquetas **a's** será comparado con la **b** existente, si es una **b** válida, se reproduce, si no, se limpia el buffer de todas las **b's** no válidas.
15. Al reproducir una **b**, si la **b** de la última **a** reproducida es menor o igual a la de la **b** actual, se acepta, en otro caso, se descarta.
16. Las **b** nuevas pueden llegar por el propio flujo de las **b's**, o por el flujo de las **a's**.
17. El LDU que llegue a tiempo, resincronizará a tiempo real a la comunicación y discriminará a los ULDs que lleguen retrasados en todos los flujos.
18. Si un LDU que debería ser reproducido no ha sido aún recibido, el receptor reproducirá un LDU predeterminado de relleno, este LDU depende del tipo de flujo manejado. El mecanismo de reproducción guarda memoria para saber si el último LDU reproducido es de relleno o es real.
19. En caso de que el siguiente LDU a reproducir sea el LDU_i y ya haya sido reproducido el LDU_{i+1} real (no de relleno), el LDU_i es desechado, en otro caso, es reproducido.
20. El algoritmo puede adaptarse a cambios en la frecuencia de producción de los dispositivos en comunicación, es decir puede cambiar el etiquetado de ULDs en tiempo de ejecución.
21. En caso de llegar a la cota inferior de CdS, definida por el usuario, algoritmo notifica al protocolo de negociación (módulo externo) de este evento para que intente solucionarlo y espera un tiempo finito para que ese problema sea solucionado. En caso de que no suceda, notifica de la imposibilidad de transmitir manteniendo una CdS y deja de transmitir ese flujo. Se deja como opción que la aplicación superior pueda suspender la sesión completa.
22. Cuando el algoritmo decide callar a uno de sus flujos, avisa al sistema general para que este decida si se termina la sesión o permite que continúen el resto de los flujos. De esta manera se generaliza el funcionamiento, por *ejemplo videoconferencia vs. Video por demanda*. En la primera, podemos continuar sin alguno de los medios, mientras que en la segunda, si falta alguno de los flujos, la sesión completa es inútil al usuario.
23. El algoritmo se adapta al inicio defasado de los objetos multimedia, esto mediante la transmisión de ULDs vacíos. De esta manera se simula que todos los flujos inician a enviar al mismo tiempo. Esto se muestra en la figura 7.2.

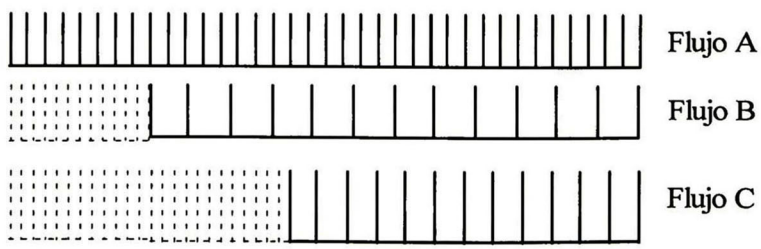


Fig. 7.2
Inicio defasado

24. Buffering. El valor inicial del buffer en el receptor, es dado al momento de instanciar el módulo. Este módulo lo verá como un valor a priori, mientras que la aplicación de manejo podría cambiarla antes de instanciar al proceso.
25. Existen algunos medios que requieren ser enviados por un canal confiable, por ejemplo: texto en un chat, títulos en un video por demanda, slides en un slideshow. La decisión de cuáles medios van por un canal confiable y cuáles no, es hecha por el implementador, pero este puede ceder la responsabilidad al usuario. El criterio de decisión puede ser el siguiente: una relación entre ancho de banda disponible y ancho de banda requerido.

    RBW / ABW
    RBW = Ancho de banda requerido
    ABW = Ancho de banda disponible

    Y este debe ser, como sugerencia $\cong$ 1/20. Un cociente mayor devendría en un posible fallo en la Calidad de Servicio [48].

7.1.4 Sincronización Intraflujo

La sincronización Intraflujo se hace por medio del cálculo de la media estimada de los tiempos interarribo de un flujo. Si se estima que un paquete tarde un tiempo **T** en llegar a su destino, entonces se hace que los paquetes sean enviados con frecuencia **T** para evitar pérdidas en los espacios intermedios.

7.1.5 Etiquetado

Como ya se mencionó anteriormente, el problema de la sincronización Interflujo es resuelta mediante el etiquetado de los ULDs.
Las características de etiquetado son enumeradas a continuación.

1. Todos los ULDs de un mismo flujo serán etiquetados con un identificador distinto y no decreciente, esto es:
   Sea ID_i el identificador de LDU_i, y
   Sea ID_j el identificador de LDU_j.
   Si $t(LDU_i) < t(LDU_j)$ entonces $ID_i < ID_j$ y
   Si $t(LDU_i) \geq t(LDU_j)$ entonces $ID_i \geq ID_j$
2. El flujo que tiene λ menor, tiene más posibilidades de llegar, ya que tiene una mayor frecuencia de envío por la red. Debido a esto, es el indicado para tener las etiquetas de los demás, es decir, es quien lleva la pauta para la sincronización Interflujo del resto.
3. En [48] se muestra que la sincronización Interflujo tiene la propiedad transitiva, y por tanto es suficiente sincronizar los flujos por pares para que haya sincronización entre todos ellos. Pero es necesario tener redundancia (solo la necesaria) para aumentar la probabilidad de sincronización. Se tienen cuatro posibilidades de etiquetado:

	1	*2*	*3*	*4*
A	ABC	AB	ABC	ABC
B	B	BC	ABC	BC
C	C	C	ABC	B

Tabla 7.1
Ejemplo de etiquetado

La opción 1: si por algún motivo se retrasara mucho el flujo A o se perdiese, los flujos B y C no tendrían forma de sincronizarse.
La opción 2: si se perdiera o se retrasara mucho el flujo B, los flujos A y C no podrían sincronizarse.
La opción 3: hay sincronización en todos los casos, pero se redunda en los datos de sincronización.
La opción 4: es una solución intermedia pues no tiene tanta información como la opción 3, y aunque se perdiera cualquiera de los flujos, la sincronización sería posible entre los otros dos.

4. Los flujos plesiócronos serán tratados de manera especial. Tendrán una etiqueta y un λ infinito. El flujo de λ menor incluirá la etiqueta del flujo plesiócrono. El flujo plesiócrono será reproducido hasta que se reciba una etiqueta igual a la suya. Esta solución es debida a que un flujo plesiócrono puede adelantarse en la reproducción, no por motivos de la red, sino de los buffers. Esto se ilustra en la figura 7.3.

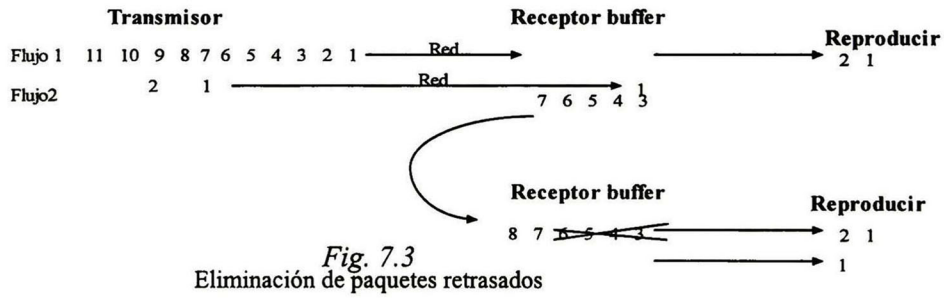


Fig. 7.3
Eliminación de paquetes retrasados

5. Las etiquetas son extraídas como se ve en la figura 7.4.

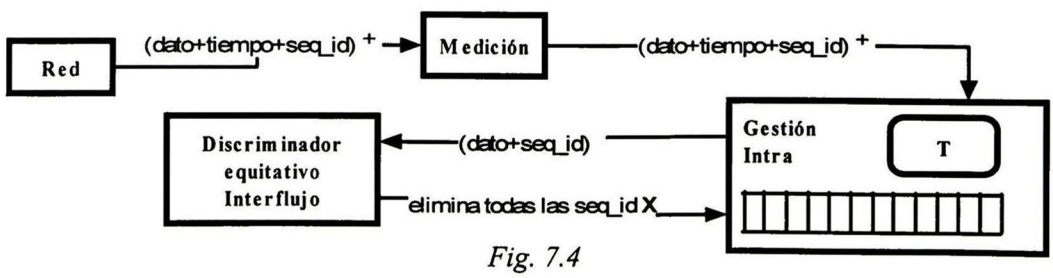


Fig. 7.4
Extracción de etiquetado

7.1.6 Arquitectura Abstracta

La arquitectura de este algoritmo está diseñada como un bloque independiente de cualquier sistema que lo contenga, logrando de esta manera que pueda ser integrado a varios tipos de implementación.

En principio, son entidades simples, solo transmiten o solo reciben, siendo estas dos acciones excluyentes, cumpliendo así con nuestra definición de Sesión. Esto se muestra en la figura 7.5.

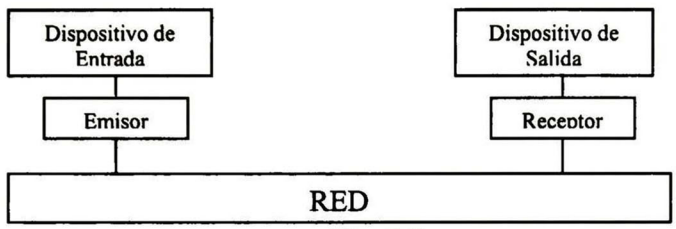


Fig. 7.5
Dos entidades simples

Aunque en la implementación real, la misma terminal física puede tener, y de hecho tendrá, los dos comportamientos. Esto se ilustra en la figura 7.6.

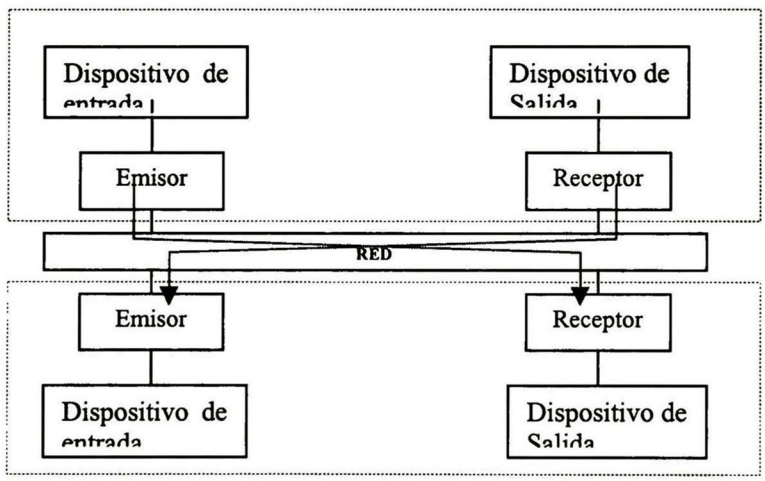


Fig. 7.6
Dos entidades compuestas

7.1.7 Arquitectura Modular

7.1.7.1 Arquitectura del Emisor

La figura 7.7 muestra la estructura interna en detalle de la entidad emisora.

El elemento común para todos los objetos multimedia es el **Control Interflujo**, cuya función es etiquetar cada paquete de red de la manera descrita anteriormente. Para cada

objeto multimedia se requiere una **Interfaz de Red Confiable, Interfaz de Red No Confiable, Gestiona, Captura e Informa.** Los módulos **Gestiona** y **Captura** son los encargados de la sincronización Intraflujo, ellos cambian la frecuencia de envío de paquetes.

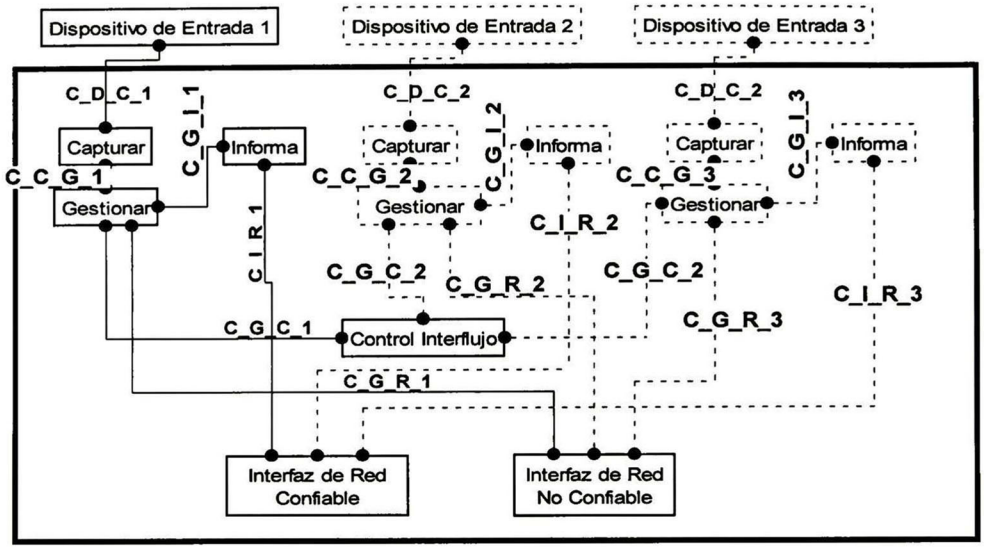


Fig. 7.7
Entidad transmisora

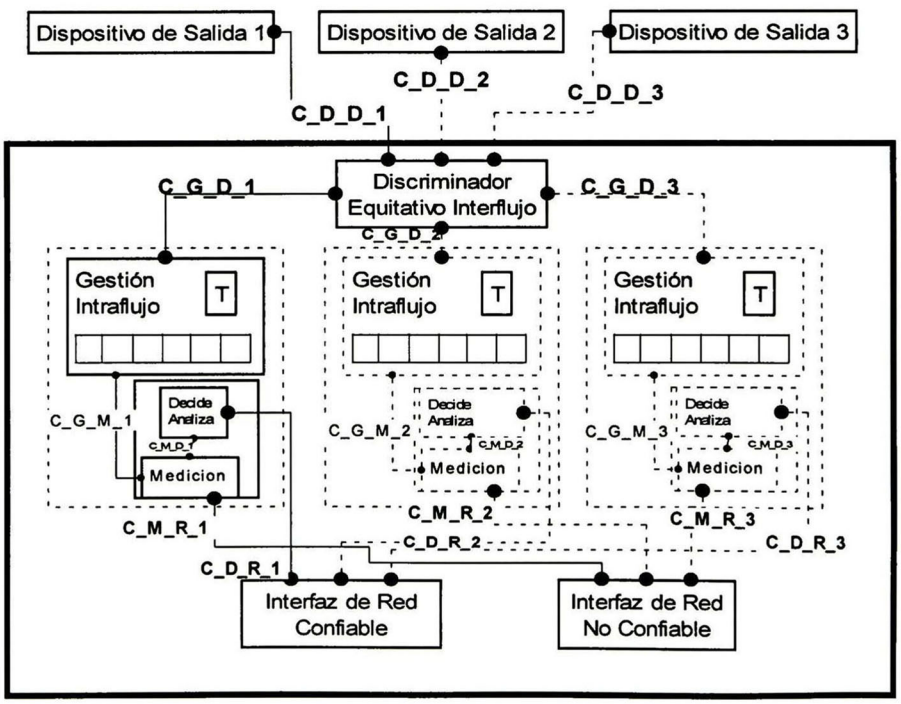


Fig. 7.8
Entidad receptora

7.1.7.2 Arquitectura del Receptor

La figura 7.8 refleja la estructura interna detallada de la entidad receptora.

En este esquema existe un módulo **Discriminador Equitativo Interflujo** común para todos los objetos multimedia. Este módulo es el encargado de la sincronización Interflujo pudiendo descartar ULDs que llegan con retraso.
Existe un módulo **Decide – Analiza, Medición, Gestión Intraflujo, Interfaz de Red Confiable e Interfaz de Red No Confiable** para cada objeto multimedia. Entre estos elementos se realiza la sincronización Intraflujo.

7.1.8 Pseudocódigo

El algoritmo se basa en la medición de tiempos de interarribo de paquetes de red en una ventana de tiempo, cada paquete de red puede contener varios ULDs. Con estos datos, se calcula un estimado del tiempo más probable de arribo del siguiente paquete. Si este tiempo es *suficientemente distinto* [48] al estimado anterior, se notifica a la entidad transmisora para que modifique el tiempo de envío de paquetes. Si la frecuencia de envío es menor, el tamaño de paquete aumenta, y si la frecuencia es mayor, el tamaño de paquete disminuye.

7.1.8.1 Primera aproximación al algoritmo de sincronización Intraflujo.

```
Var primer1 = true, primer2 = true

1.-    si primer1 = true
              primer1 = false
              ve al paso 2
       si no
              calcula tamaño de ventana
              si ventana > ventana actual
                           cambia ventana

2.-    Recibe tiempo de interarribo

3.-    Guarda en ventana

4.-    Si la ventana no está llena
              ve al paso 2

5.-    Calcula  μ   σ²

7.1.-  Si primer2 = true
              ve al paso 10

7..-   Calcula likelyhood ratio

8.-    Decide sobre likelyhood ratio

9.-    Si es necesario
```

cambia $\hat{\mu}$ $\hat{\sigma}^2$ por las nuevas

10.- Calcula la función erf()
primer2 = false

11.- Si es necesario
notifica nueva frecuencia

12.- Vacía ventana

13.- Ve al paso 1

7.1.8.2 Refinamiento del algoritmo.

Vars
v = 30,w 30 :enteros
primer1 = true, primer2 = true : booleanos
ventana = arreglo[w * 2] de reales
$\hat{\mu}$ $\hat{\sigma}^2$ μ σ^2 reales
timer = 0

1.- Si primer1 = true
ve al paso 2
si no
v = calcula nuevo tamaño en base a μ y σ^2
si v > (w * 2)
destruye ventana
ventana = crea un arreglo [v * 2] de reales
w = v

2.- Aguarda a recibir un NDU

3.- Si primer1 = falso
ventana[i] valor del timer
i i + 1
reinicia el timer a 0
si no
primer1 = falso
reinicia timer a 0

4.- Si i < w
ve al paso 2

5.- $$\hat{\mu} = \frac{1}{w}\sum_{j=0}^{w-1} ventana[j]$$

$$\hat{\sigma}^2 = \frac{1}{w-1}\sum_{j=0}^{w-1} (\hat{\mu} - ventana[j])^2$$

6.- Si primer2 = true
$$\mu = \hat{\mu}$$
$$\sigma^2 = \hat{\sigma}^2$$
ve al paso 10

```
7.-   Calcula likelyhood ratio

8.-   Si lratio > k
            μ = μ̂
            σ² = σ̂²
            notificar   true
      si no
            notificar   false

9.-   Pnueva   tiempo con mayor probabilidad de arribo (erf)
      primer2   false

10.-  Si notificar   true
      notificar Pnueva al transmisor

11.-  i = 0

12.-  ve al paso 1
```

7.2 Arquitectura Base

7.2.1 Introducción

Entendemos por arquitectura de software a la estructura lógica construida sobre:

- Protocolos de comunicación
- Diagramas de módulos de organización de protocolos
- Diagramas de capas de organización de protocolos
- Diagramas de módulos de la relación entre subsistemas

Primero se muestran las modificaciones hechas a la recomendación H.323 y después la arquitectura propuesta, tomando como base los modelos de la especificación H.323. En la sección 7.2.2 encontramos los modelos de H.323 y las modificaciones propuestas para la agregación de un algoritmo de sincronización, esto visto como módulos. En la sección 7.2.3 encontramos la organización en capas de los protocolos de H.323 con el anexo del algoritmo de sincronización, visto como diagrama de bloques. La arquitectura de software que implementa H.323, los módulos de sincronización, bases de datos, control e interfaces, son mostrados en la sección 7.2.4. la sección 7.2.5 muestra los cronogramas de comunicación entre elementos del sistema global descrito en el capítulo 4; mientras que la sección 7.2.6 describe algunas consideraciones hechas sobre esta arquitectura.

7.2.2 Modificación a H.323 en el modelo de capas

Como ya se ha explicado, ITU TH.323 es una recomendación *"sombrilla"*, esto es, que cubre a varias otras recomendaciones. En la figura 7.9 encontramos un primer diagrama donde se muestran las relaciones que existen entre los diferentes elementos de esta recomendación mediante un modelo a capas.

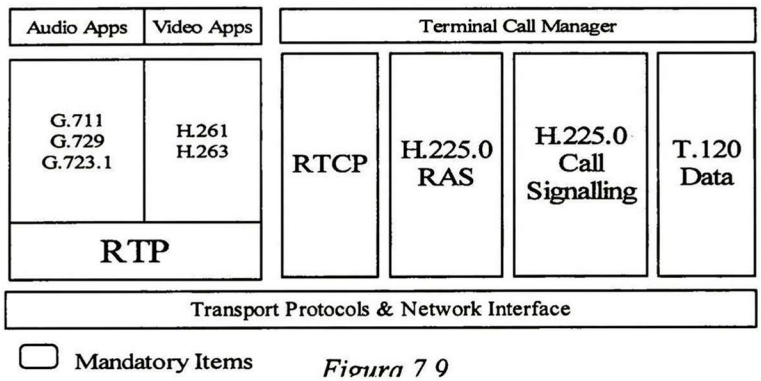


Figura 7.9
Modelo original H.323

En esta figura están sombreado de gris los ítems que son obligatorios en H.323, aunque estamos hablando solamente de protocolos, ya que H.323 define que como servicio básico debe existir la comunicación de audio, mientras que el resto de medios, son opcionales.

El algoritmo escogido no tiene capacidades de controlar la creación y destrucción de canales lógicos, pero requiere de canales confiables para la transmisión de la información de control interna al algoritmo, además de los canales no confiables para la transmisión de los datos multimedia [Sección 6.1]. En la figura 7.10 podemos ver la posición del algoritmo dentro de esta primera vista de la recomendación H.323

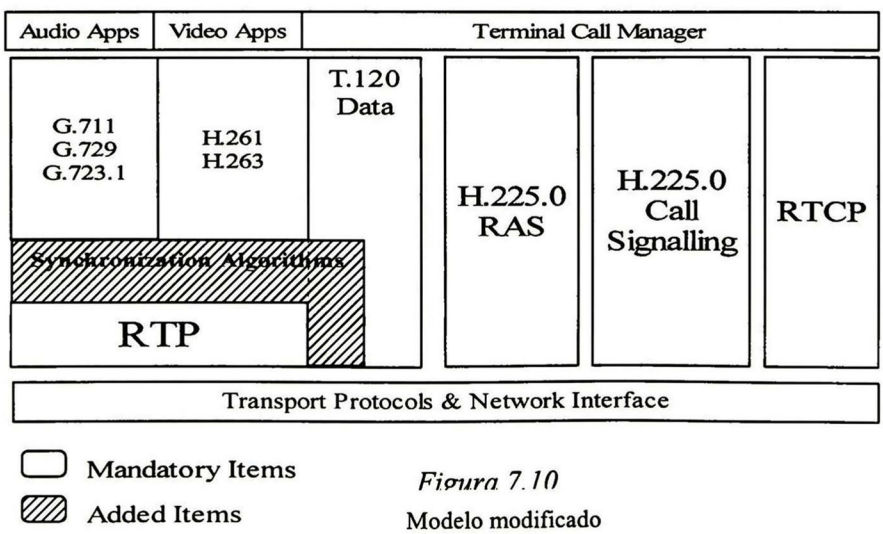


Figura 7.10
Modelo modificado

Si hiciéramos una expansión de los bloques mostrados, veríamos que la recomendación T.120 es en realidad una serie de recomendaciones de las cuales solo tres, la utilizadas por H.323, son obligatorias. También veríamos que la capa de transporte mostrada en la gráfica anterior, está dividida en varias. Esta nueva vista de los elementos de H.323 es mostrada en la figura 7.11 junto con el anexo del algoritmo y su relación con el resto de los elementos

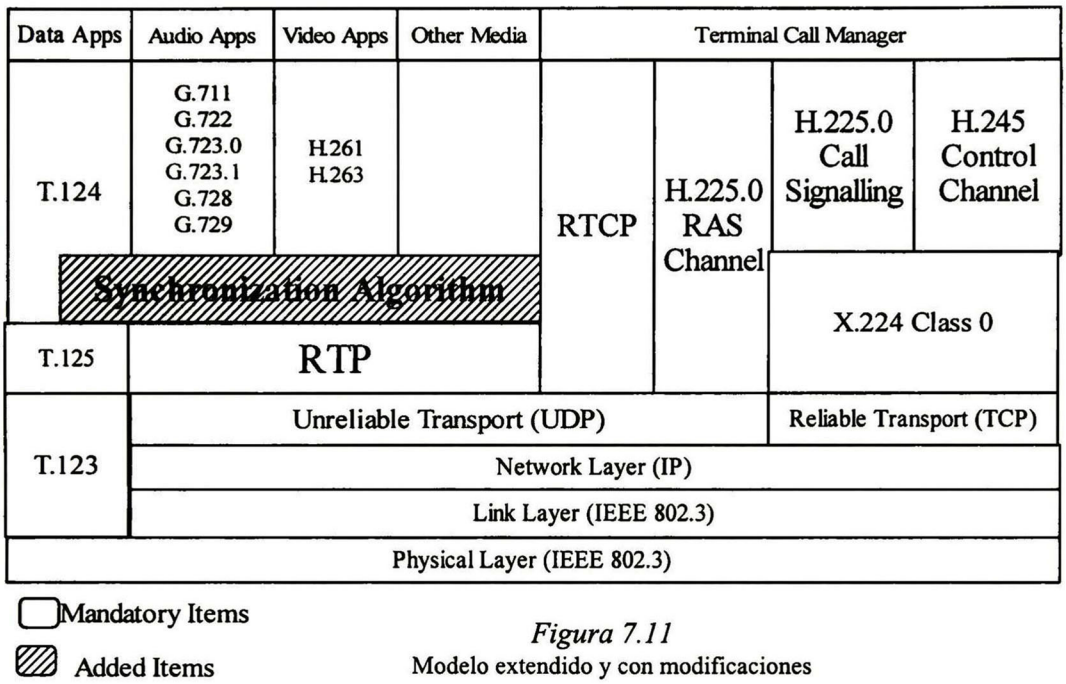


Figura 7.11
Modelo extendido y con modificaciones

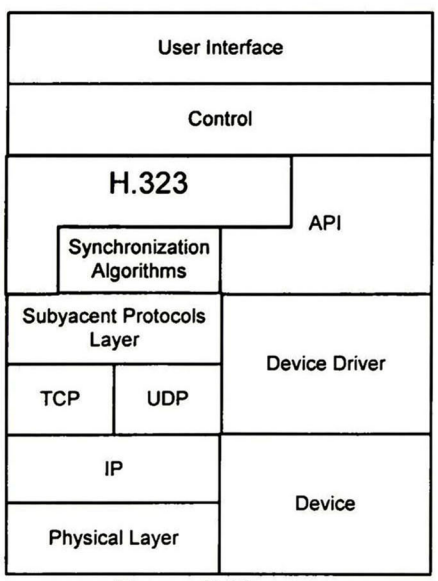


Figura 7.12
Ubicación en el sistema

Los diagramas anteriores muestran nuestra arquitectura como una modificación a H.323, pero también debemos integrar las interfaces con los dispositivos y servicios especificados en el capítulo 4.

La figura 7.12 muestra la ubicación de la recomendación H.323 y el algoritmo escogido, con respecto al resto del sistema. En esta figura podemos ver que H.323 es utilizado por el control como una capa de servicio de comunicaciones y que el control no se da cuenta de los algoritmos de sincronización. También podemos observar que esta capa de control, junto con la de sincronización

y H.323, pueden ser utilizadas como un solo bloque por diferentes aplicaciones. En esta figura tenemos el bloque H.323 separado del bloque de sincronización, pero como hemos visto en los diagramas anteriores, en realidad están mezclados. También podemos observar un pequeño bloque obscuro debajo del bloque H.323, esto es porque el bloque de sincronización no cubre la comunicación completa, sino solo las relacionadas con los medios, aunque hace uso de algunos canales confiables para la retroalimentación del estimado de la red.

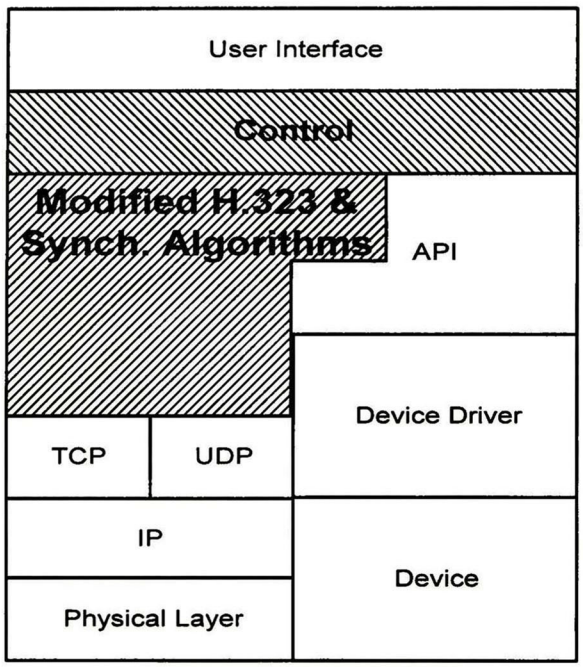


Figura 7.13
Bloque independiente

Podemos ver también que tanto el control como el bloque de H.323 tienen contacto con el controlador de los dispositivos. Esto es debido a que el control es quien tiene el primer contacto con los medios y puede permitir a H.323 y los codecs incluidos en esta recomendación, que manejen la información que se recibe de los medios.

También podemos ver como el bloque de sincronización es independiente de la forma de codificación de los medios, de la fuente de los medios y de los medios de transmisión, aunque esto se observa mejor en las figuras 7.15 y 7.16.

Si agrupáramos el bloque H.323 con el bloque de sincronización y los señaláramos junto con la etapa del control del sistema, podríamos tomarlo como un bloque único. Esto nos daría la ventaja de poder cambiarlo de aplicación y de medios sin afectar grandemente al resto del sistema. Esto lo podemos ver en la figura 7.13.

7.2.3 Modificación a H.323 en el modelo de módulos

El esquema original propuesto por la ITU se muestra en la figura 7.14. En esta podemos ver por donde pasan la información de los dispositivos multimedia y por donde los flujos de control. Además podemos ubicar el elemento de control principal, aunque la ITU no define su arquitectura. En nuestro caso representaremos el comportamiento del bloque de control, mediante Máquinas de Estado Finito (FSM).

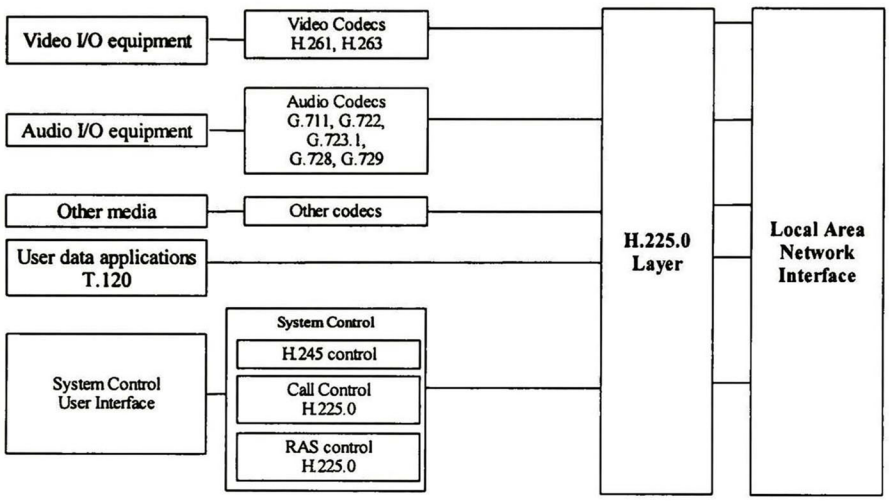


Figura 7.14
Esquema original H.323

Este esquema es planteado tanto para el transmisor como para el receptor.

La figura 7.15 muestra la modificación propuesta de esta arquitectura para el lado transmisor. En esta figura podemos ver un elemento extra que asigna etiquetas de flujo para lograr la sincronización Interflujo además de un administrador de paquetes que cambia la frecuencia, y por tanto el tamaño, de los paquetes enviados.

En la figura 7.16 se muestra la arquitectura del receptor, donde se ve que tiene más modificaciones que el lado transmisor. Es este lado de la comunicación el que será responsable de hacer un estimado de la red y de desechar ULDs.

Una terminal de comunicación contendrá a ambas, la parte de transmisión, como la parte de recepción. La figura 7.17 muestra una arquitectura donde se unen estas dos partes.

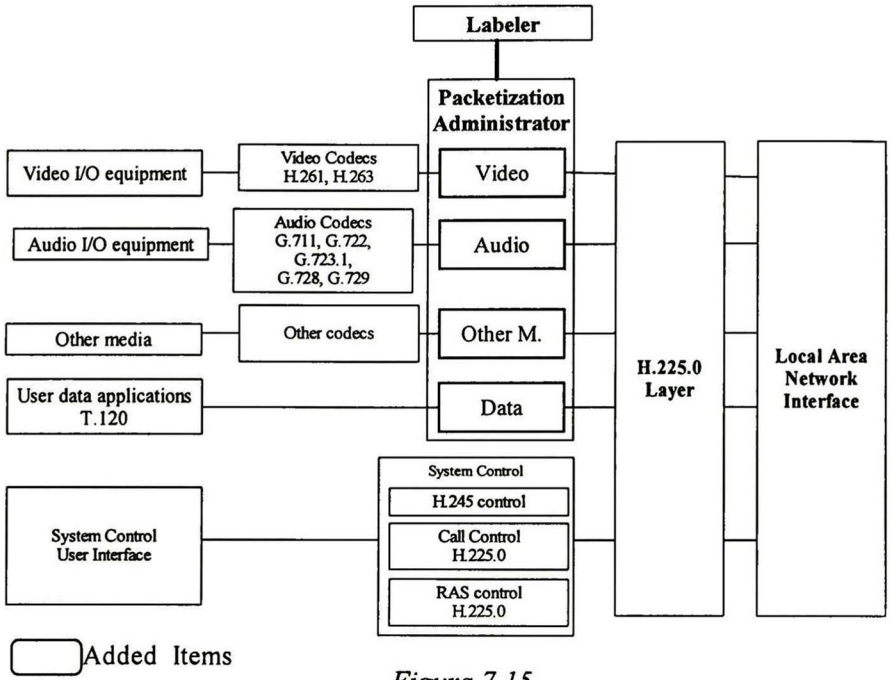


Figura 7.15

Esquema modificado para el transmisor

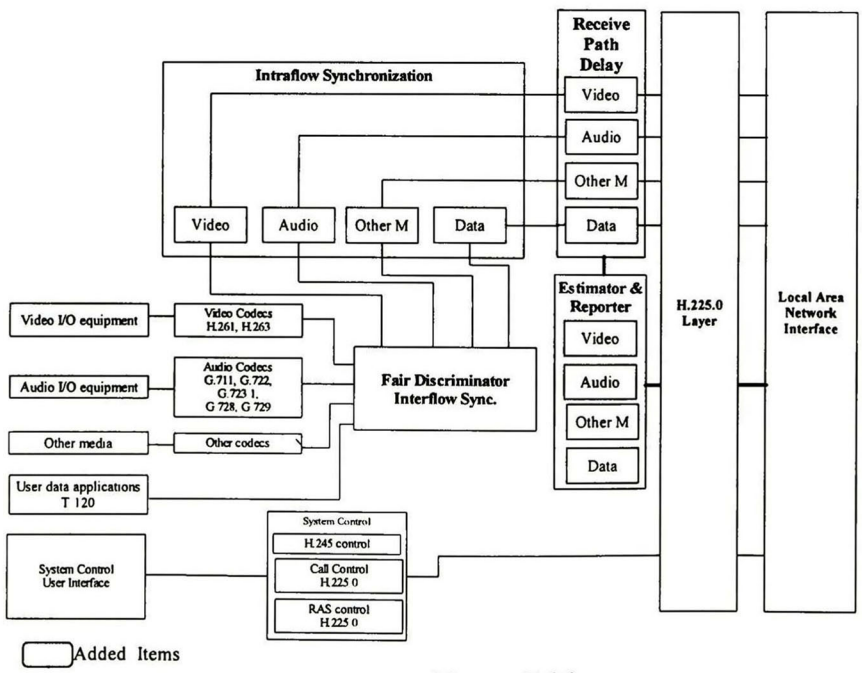


Figura 7.16

Esquema modificado para el receptor

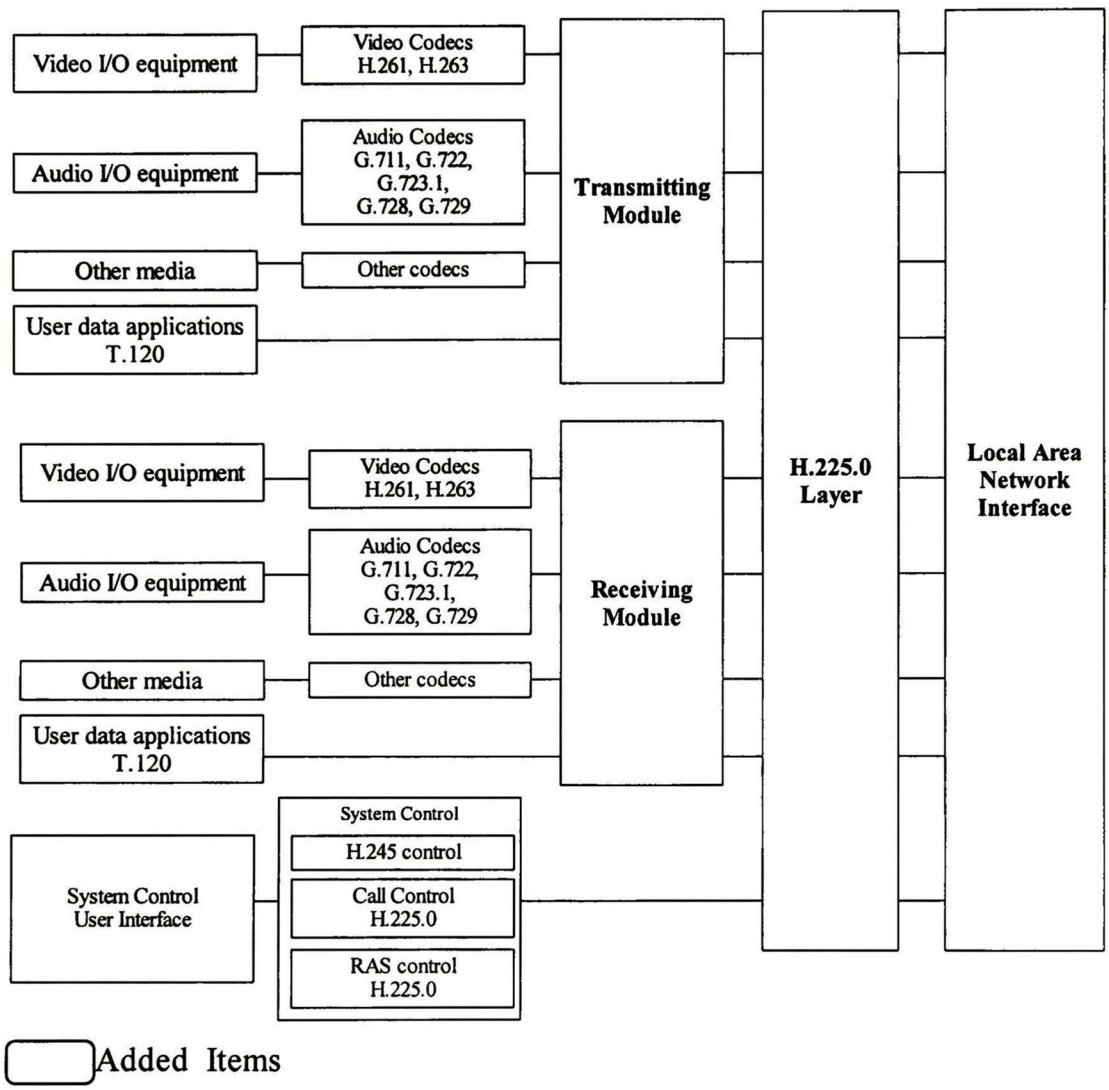


Figura 7.17
Esquema de Transmisión - Recepción

7.2.4 Arquitectura de Software Propuesta

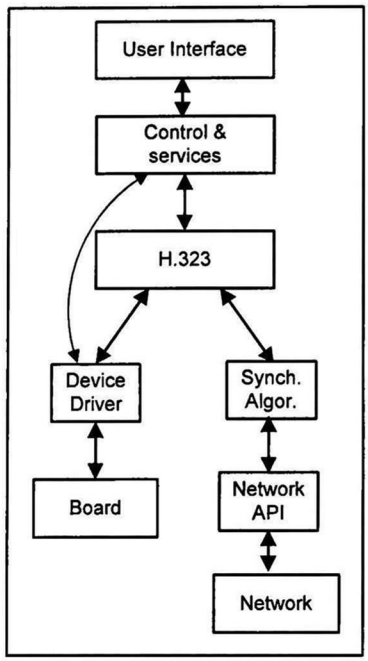


Figura 7.18
Interacción con dispositivos

Como ya mencionamos anteriormente, esta modificación hecha a la recomendación H.323, debe ser unida a una arquitectura de software que maneje los canales y los dispositivos, además de interactuar con el software de aplicación. La figura 7.18 nos muestra un diagrama de bloques donde se describe esto.

En esta no esta muy claro el flujo de información entre los diferentes elementos del software, pero es una primera aproximación que iremos ampliando en diagramas posteriores.

La figura 7.19 nos muestra otra vista de la arquitectura propuesta. Aquí podemos ver una base de datos (especificada en el capítulo 4 de esta tesis) en la que se almacena la información de directorio, identidad y configuración y que es local a la terminal. Podríamos pensar en tener una base de datos distribuida, pero esto sería parte del gatekeeper.

También podemos observar que la interfaz con el usuario no está siendo considerada como parte de esta arquitectura de software. Esto es también explicado en las figuras 7.12 y 7.13.

Esta figura7.19 muestra tres grandes módulos principales, *Network Management, Control* y *Device Management.* La recomendación H.323 y el bloque de sincronización se encuentran distribuidas entre los módulos *Network Management* y *Device Management*, mientras que el módulo de *Control* es el motor de la comunicación y punto de unión entre todos los protocolos.

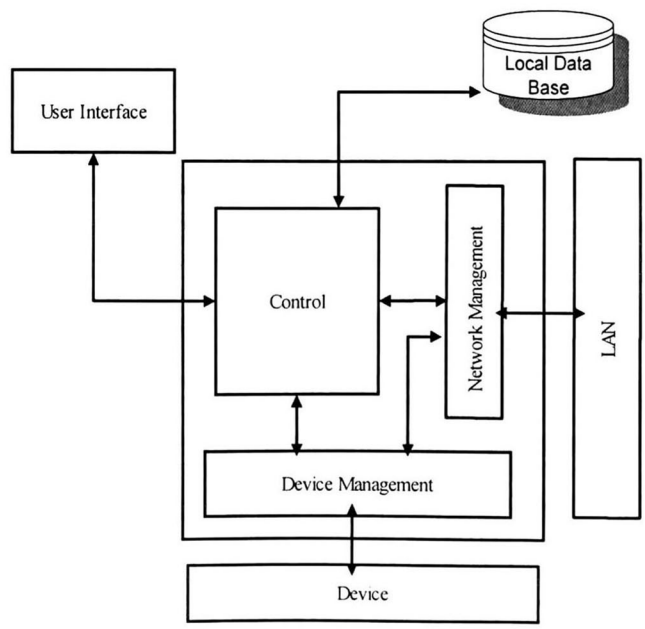


Figura 7.19
Arquitectura inicial

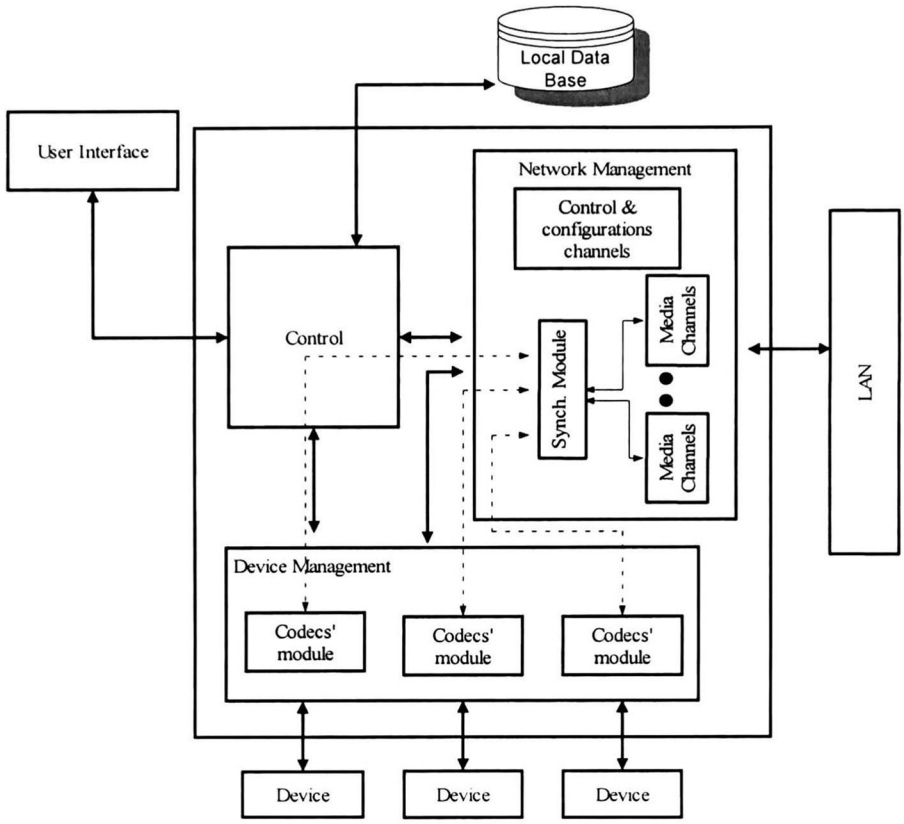


Figura 7.20
Esquema para varios medios

Dentro del módulo *Network Management* podemos encontrar la parte de H.323 que se refiere a los protocolos de comunicación (H.245, H.225, T.120, RAS, etc.) y el bloque de sincronización agregado por nosotros a H.323. Como ya se explicó en capítulos anteriores, existe un solo canal de control por una sesión, mientras que pueden existir varios canales de comunicación de medios, uno por cada medio que se desee transmitir y por cada dirección. Debemos recordar también que una sesión es una comunicación punto a punto entre dos terminales.

En el bloque *Device Management* encontramos la interacción con la API de programación de los dispositivos de medios, además de los codificadores para los medios. Estos los podemos tratar de manera separada al bloque de comunicación en red para hacer un código más reutilizable.

Esta estructura de comunicación para una sola sesión la podemos observar en la figura 7.20

En la figura 7.21 podemos observar la estructura que tendría el sistema en el caso de varias sesiones. Podemos tener un solo bloque de control que crea varias instancias de bloques *Network Management-Device Management*, una instancia para cada sesión.

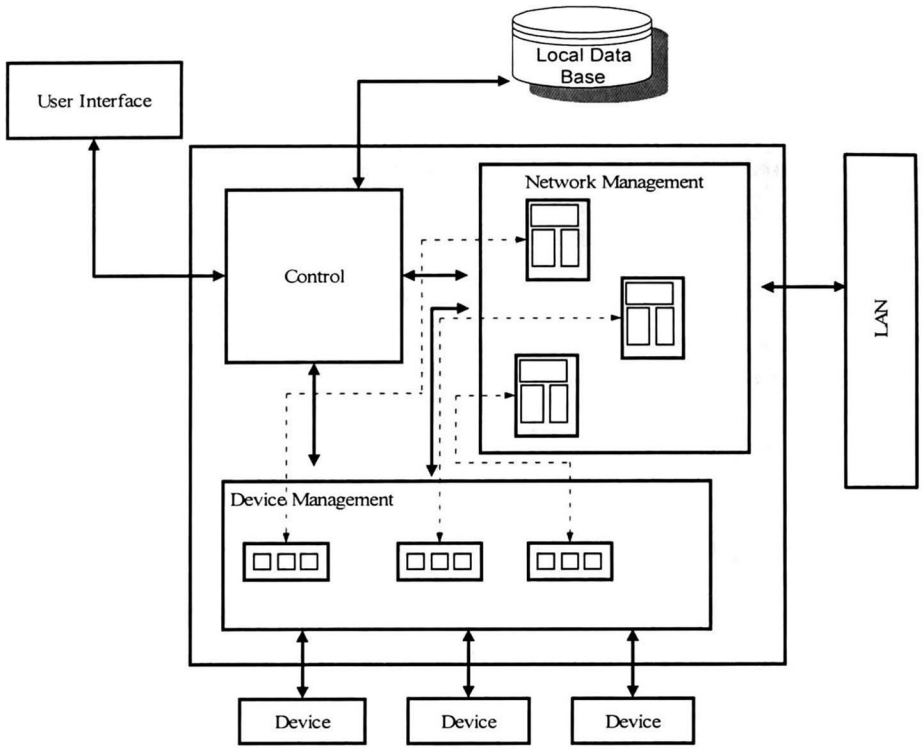


Figura 7.21
Esquema de múltiples sesiones

7.2.5 Conexión con tarjetas telefónicas.

Como se mencionó en el capítulo 4, el sistema debe poder tomar datos de las tarjetas *IP/4 Port Trunk card* y *IP/6 Port FXS* card además de *IP/ G.723.1 compression card* [capítulo 4].

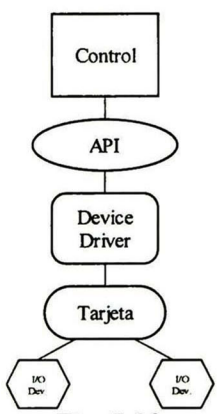


Fig. 7.22
Interacción con tarjetas

En esta sección mostraremos los cronogramas creados para especificar las comunicaciones necesarias para la conexión de este sistema con las tarjetas mencionadas.

Como acuerdo, toda la arquitectura mostrada anteriormente(control, device management, network management y data base), será llamada simplemente **Control** en los cronogramas y esta interactuará con una API de programación del dispositivo y un Device Driver, esto se puede ver en la figura 7.22.

En los primeros cronogramas, se toma como dispositivo de entrada/salida a un teléfono o unas bocinas y un micrófono, mientras que la **Terminal** está compuesta por *tarjeta, device driver, API y control.*

7.2.5.1 Cronogramas de comunicación Terminal – Terminal

En la figura 7.23 podemos ver todas las comunicaciones necesarias para un enlace exitoso mediante el protocolo H.323.

Primero, la terminal solicitante se comunica con el Gatekeeper para solicitar restricciones (prefijos restringidos, permisos de comunicación, terminales restringidas) y reservar ancho de banda. Si los permisos son concedidos, solicita comunicación con la otra terminal. La terminal receptora, también debe solicitar permisos al Gatekeeper. Estas solicitudes de permisos al Gatekeeper son hechas mediante el protocolo RAS (ver capítulo 4). Si los permisos son concedidos, comienza la etapa de intercambio de capacidades (velocidad de transmisión, cantidad y tipo de medios, tipos de codificadores), esto se hace con el protocolo H.245. una vez abiertos los canales de comunicación (tipo RTP) comienza la transmisión de información multimedia. El final de la sesión, se hace con los protocolos H.245 y H.225 entre las terminales y el protocolo RAS para notificar al Gatekeeper del término de la sesión, de esta manera, el Gatekeeper podrá utilizar ese ancho de banda para otra comunicación.

7.2.5.2 Cronogramas de comunicación Terminal – Terminal con interfaz a dispositivos E/S

Como podemos ver en la figura 7.24, las comunicaciones entre las terminales y el Gatekeeper, son las mismas (estas deben mantenerse para que el sistema sea compatible con el estándar H.323), pero ahora podemos ver un dispositivo I/O. Para nuestro sistema, este dispositivo I/O puede ser un equipo multimedia (bocinas y micrófono), un sistema telefónico o un archivo. En diagramas posteriores podremos ver las comunicaciones necesarias para los casos particulares de las tarjetas FXS y de troncales (cap. 4).

En esta figura 7.24 tenemos el caso particular de un teléfono como dispositivo I/O. Podemos observar los procesos de descolgado y colgado del auricular *(off hook y on hook* respectivamente). Podemos ver también los procesos de transmisión de tonos, generación de ring, ocupado y ring back (*DTMF, Ring, Busy, Ring Back* respectivamente).

En la misma figura podemos ver el procedimiento de descolgado, conexión, respuesta, transmisión, colgado y desconexión en este mismo orden, y esto, siendo compatible con el estándar H.323.

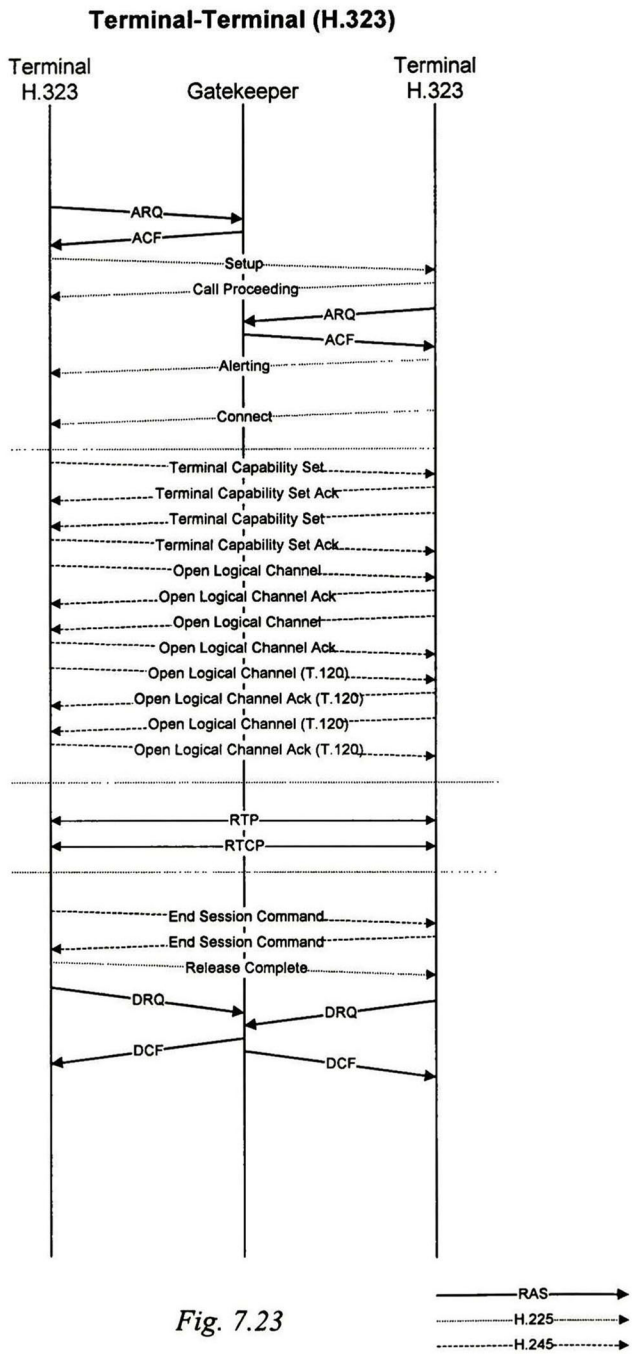

Terminal-Terminal (H.323)
Terminal H.323
Gatekeeper
Terminal H.323
ARQ
ACF
Setup
Call Proceeding
ARQ
ACF
Alerting
Connect
Terminal Capability Set
Terminal Capability Set Ack
Terminal Capability Set
Terminal Capability Set Ack
Open Logical Channel
Open Logical Channel Ack
Open Logical Channel
Open Logical Channel Ack
Open Logical Channel (T.120)
Open Logical Channel Ack (T.120)
Open Logical Channel (T.120)
Open Logical Channel Ack (T.120)
RTP
RTCP
End Session Command
End Session Command
Release Complete
DRQ
DRQ
DCF
DCF
RAS
H.225
H.245

Fig. 7.23

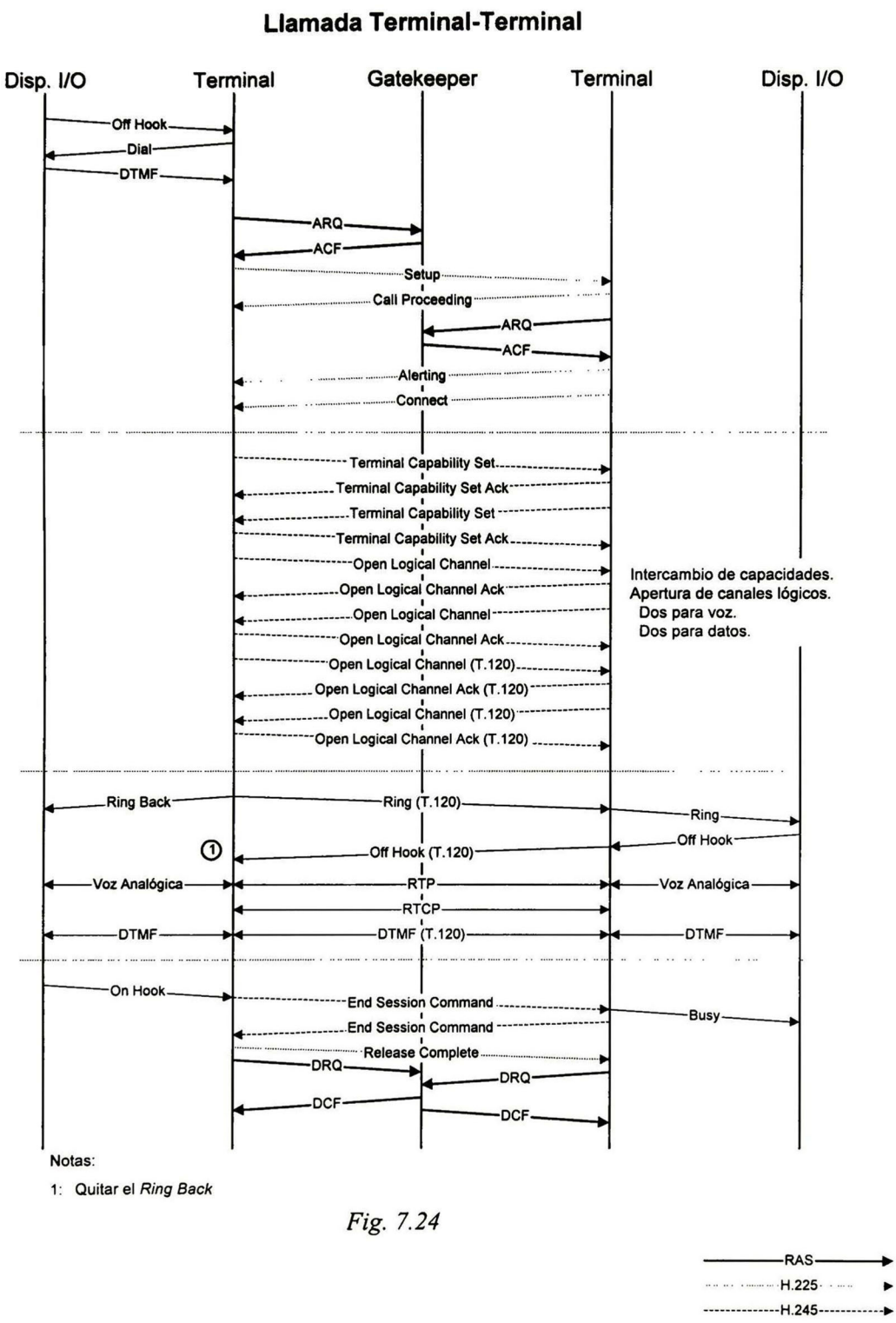

Llamada Terminal-Terminal
Disp. I/O
Terminal
Gatekeeper
Terminal
Disp. I/O
Off Hook
Dial
DTMF
ARQ
ACF
Setup
Call Proceeding
ARQ
ACF
Alerting
Connect
Terminal Capability Set
Terminal Capability Set Ack
Terminal Capability Set
Terminal Capability Set Ack
Open Logical Channel
Open Logical Channel Ack
Open Logical Channel
Open Logical Channel Ack
Open Logical Channel (T.120)
Open Logical Channel Ack (T.120)
Open Logical Channel (T.120)
Open Logical Channel Ack (T.120)
Intercambio de capacidades.
Apertura de canales lógicos.
Dos para voz.
Dos para datos.
Ring Back
Ring (T.120)
Ring
Off Hook
1
Off Hook (T.120)
Voz Analógica
RTP
Voz Analógica
RTCP
DTMF
DTMF (T.120)
DTMF
On Hook
End Session Command
Busy
End Session Command
Release Complete
DRQ
DRQ
DCF
DCF
Notas:
1: Quitar el Ring Back
RAS
H.225
H.245

Fig. 7.24

7.2.5.3 Gateway H-P

Llamamos un Gateway H-P (H.323 – PSTN) a la terminal que hace la interfaz entre la PSTN y nuestro servicio H.323. Este Gateway puede ser tratado como cualquier otra termina, y por lo mismo, debe cumplir con los protocolos de comunicación H.323 como en las figuras 7.23 y 7.24, pero la interacción hacia el exterior cambia.

7.2.5.3.1 Cronogramas de comunicación Terminal a Gateway H-P

Las figuras 7.25, 7.26, 7.27 y 7.28 muestran una comunicación entre una terminal H.323 (terminal con teléfono) y la PSTN pasando por un Gateway H-P.

En la figura 7.25 podemos observar las siguientes etapas:
a) El inicio desde una terminal, cuando se descuelga el auricular
b) La solicitud de permisos, cuando se comunica con el Gatekeeper
c) El intercambio de capacidades entre terminales
d) La asignación de una troncal a la PSTN

La figura muestra que la conexión con el Gateway H-P, es igual a la conexión con cualquier otra terminal, y que este Gateway tiene asignada una extensión como cualquier otra terminal. Para que el Gateway permita el establecimiento de la comunicación, verifica una tabla de asignación de las troncales de salida hacia PSTN, en caso de que exista una troncal disponible, la marca como ocupada y establece la comunicación con la terminal.

En esta figura se muestra que la transmisión de tonos DTMF se hará mediante un protocolo confiable (T.120). Cuando la terminal detecta que el dispositivo transmite un DTMF, bloquea la transmisión de voz y transmite una señal por T.120. El bloqueo de la voz se hace mientras la terminal transmite el DTMF, una vez que termina, se restablece la transmisión de voz. Esto se hace para evitar que se pueda repetir el tono en el lado del Gateway.

También podemos observar que la invitación a marcar (Dial) es una señal que transmite el Gateway por el protocolo T.120 a la terminal, y que la terminal tiene que generar este tono a su dispositivo I/O.

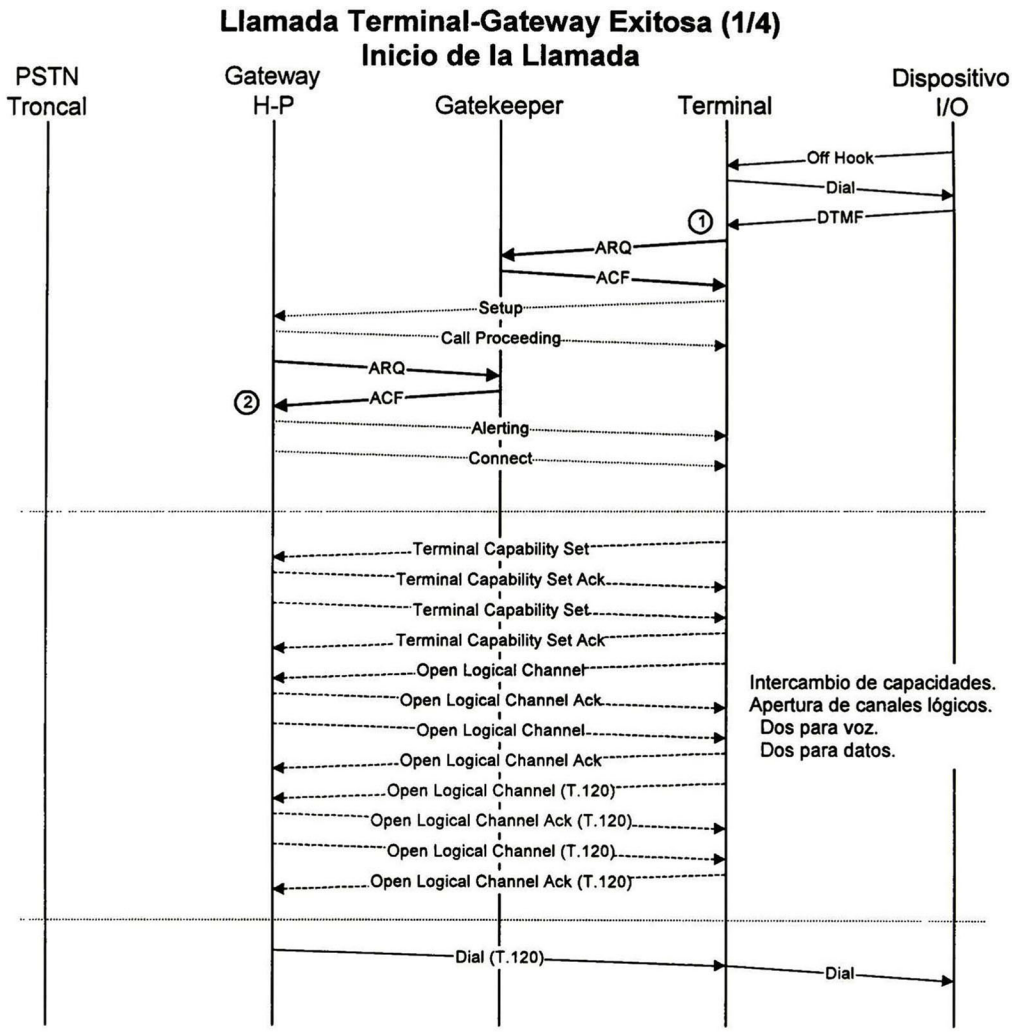
En la figura 7.26 podemos ver el proceso de enlace con una terminal en la PSTN, la terminal detecta la presencia de un DTMF desde el dispositivo I/O, bloquea la voz mientras dura el tono y transmite este por T.120, estos primeros tonos son de la clave de salida a PSTN, el Gateway valida la clave con el Gatekeeper que le regresa la validación de la clave y las restricciones de la misma (prefijos restringidos, número restringidos); el Gateway descuelga la troncal que tenía reservada y envía una señal a la terminal por T.120 para que esta genere un tono de invitación a marcar al dispositivo. La terminal comienza a recibir DTMF's y corta la generación de invitación a marcar, estos DTMF's son enviados al Gateway quien verifica si es válido con respecto a sus restricciones, si es un dígito válido, lo envía a la PSTN, de otra manera inicia el proceso de desconexión. El Gateway detecta que la llamada se ha establecido con la detección de cambio de polaridad (servicio provisto por la compañía telefónica de la PSTN). Una vez establecida la comunicación, comienza la tarificación de la llamada (billing). La voz es transmitida por RTP y los DTMF's por T.120.

En la figura 7.27 vemos el proceso de llamada de una terminal hacia la PSTN pero en el caso en que este tipo de llamadas no esta restringido. El Gateway recibe los DTMF's por T.120 y los transfiere a la PSTN sin ninguna restricción. De la misma manera que en la figura 7.26, la voz es transmitida por RTP en la red y regenerada en el Gateway para ser enviada a la PSTN, los DTMF's son transmitidos por T.120 y el Gateway los regenera hacia la PSTN.

La figura 7.28 muestra el proceso de desconexión en el que puede haber dos casos: cuelgan del lado de la PSTN o cuelgan del lado de la terminal.
En el caso de que cuelguen del lado de la PSTN, el Gateway recibirá un tono de ocupado (busy) la inversión de la polaridad de la troncal, el Gateway inicia el proceso de desconexión con la red y envía una señal de colgado a la PSTN, la terminal, al recibir el proceso de desconexión de la red, genera un tono de ocupado al dispositivo hasta que este cuelgue.

En caso de que sea la terminal la que cuelga, el proceso es un poco más sencillo. La terminal inicia el proceso de desconexión con la red, el Gateway la recibe y envía una señal de colgado a la PSTN.

En ambos casos, ya que el Gateway sabe el tiempo que la comunicación con la PSTN estuvo funcionando, es quien debe tarificar. El Gateway envía al Gatekeeper la información completa de la llamada que se realizó.

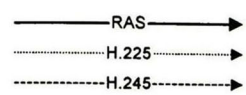


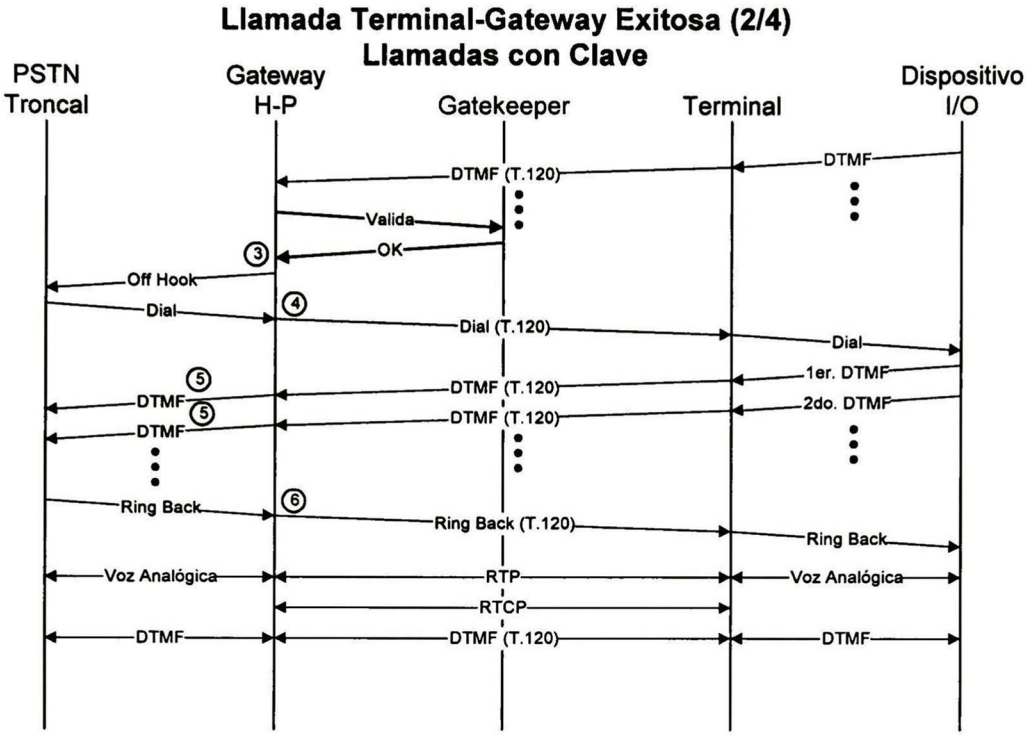
Notas:

1: Extensión del GW-HP

2: Mediante una tabla, verificar si existe línea en uso, si hay línea libre (presumiblemente) continua.
Si no, corta la llamada.
Solicita tipo de restricciones (local libre o no, prefijos restringidos). ACK (ok,0/1,00,01,...).

Fig. 7.25





Notas:

3: Recibe confirmación y restricciones asociadas a la clave. ACK(ok,00,01,...,6841580).

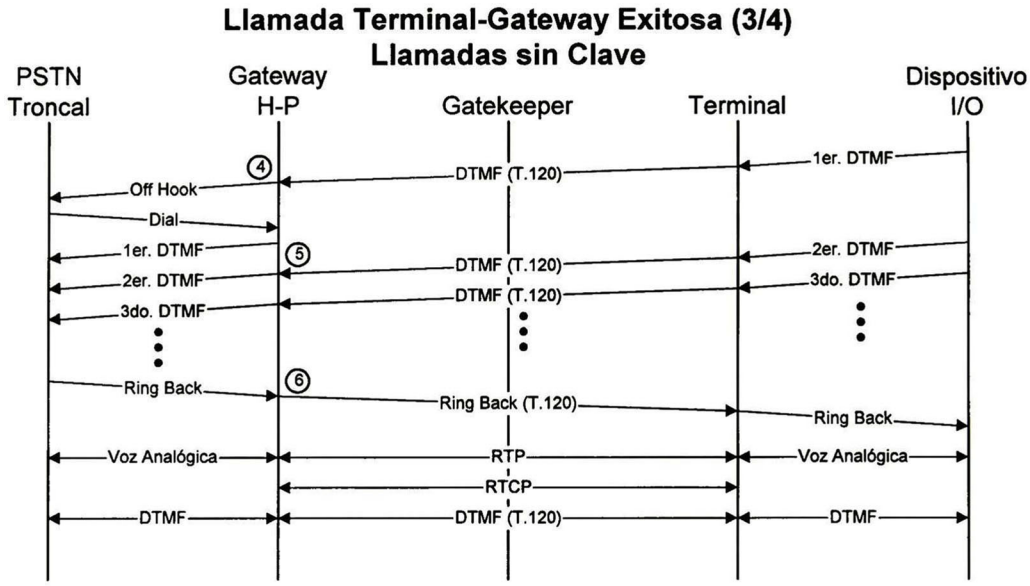
4: Polling hasta encontrar DIAL TONE, si no lo encuantra en ninguna de las líneas que se suponían libres, entonces corta la llamada.

5: Envía a la PSTN el dígito que le llega y verifica el dígito que envío, si concuerda con alguno de los patrones o prefijos prohibidos, corta la llamada.

6: Si recibe ocupado, corta la llamada. Si invierte la polaridad inicia tarificación.

RAS
H.225
H.245

Fig. 7.26



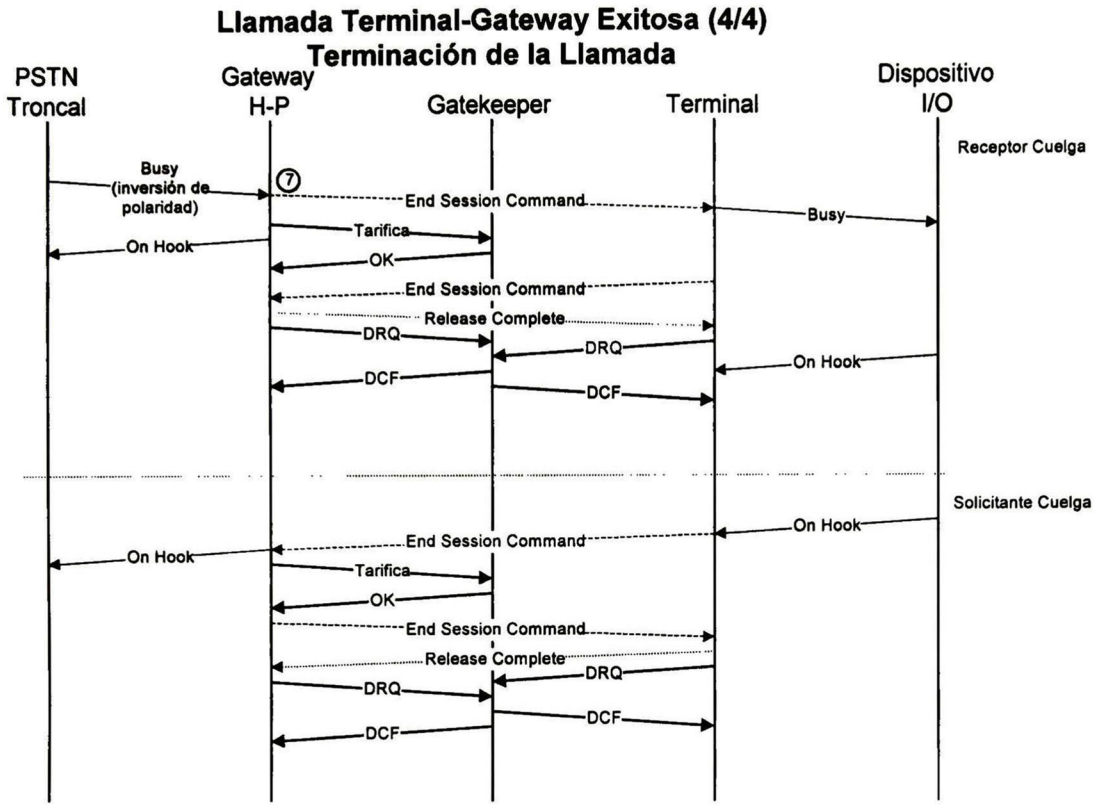
Notas:

4: Polling hasta encontrar DIAL TONE, si no lo encuantra en ninguna de las líneas que se suponían libres, entonces corta la llamada.

5: Envía a la PSTN el dígito que le llega y verifica el dígito que envío, si concuerda con alguno de los patrones o prefijos prohibidos, corta la llamada.

6: Si recibe ocupado, corta la llamada. Si invierte la polaridad inicia tarificación.

Fig. 7.27



Notas:

7: Al recibir la inversión de polaridad (la llamada termina), tarifica y cierra canales.

Fig. 7.28

7.2.5.3.2 Cronogramas de comunicación de Gateway H-P a Terminal

En una comunicación entre la PSTN y una terminal, puede ser un usuario de la PSTN quien inicie la llamada y tenga que introducirse en nuestro sistema. La secuencia de eventos necesarios para esto, es mostrada en la figura 7.29 y 7.30.

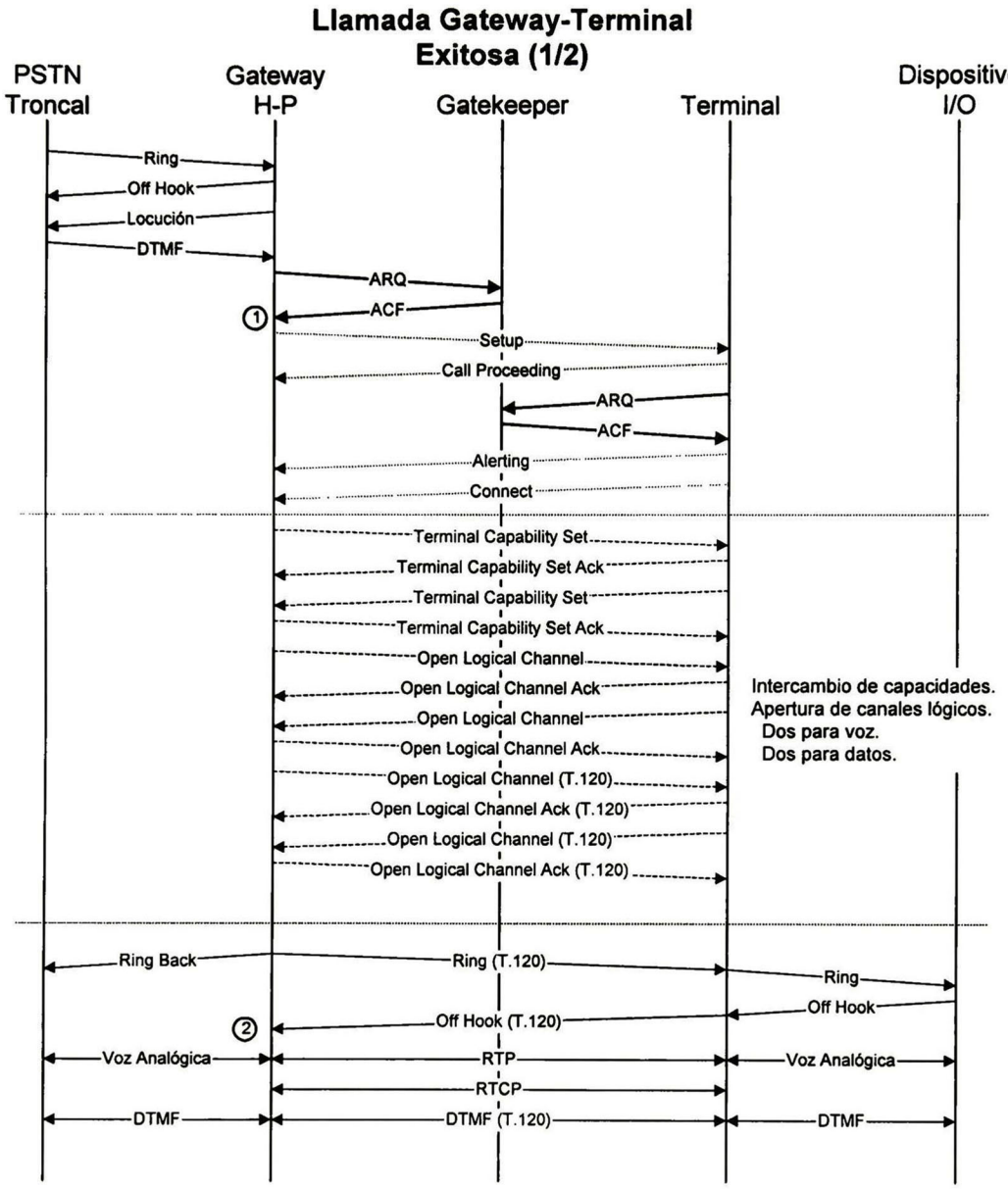
En la figura 7.29 podemos ver que el Gateway H-P debe ser capaz de detectar el ring y los DTMF's, y además debe ser capaz de generar un tono de ring back a la PSTN. Pero también podemos observar que el Gateway H-P debe generar una locución a la PSTN una vez que se haya descolgado la troncal.

Las comunicaciones entre Gateway H-P y la terminal, son las mismas que en diagramas anteriores.

La forma en que el Gateway H-P maneja los DTMF's recibidos de la PSTN, es la misma en que lo hace la terminal, al detectar un DTMF, bloquea la transmisión de voz y pasa este DTMF por una protocolo seguro (t.120).

En esta misma figura, podemos observar que una terminal debe ser capaz de generar ring a su dispositivo de I/O y debe ser capaz de generar DTMF's.

En la figura 7.30 podemos ver los casos de colgado en una comunicación Terminal – Gateway H-P. En esta comunicación, puede colgar el usuario de la PSTN o puede colgar el dispositivo de I/O. La secuencia de mensajes necesarios para la desconexión en ambos casos, es mostrada en esta figura.



Notas:

1: Regresa validación de extensión, permiso de ancho de banda y restricciones.

2: Quitar *Ring Back*

Fig. 7.29

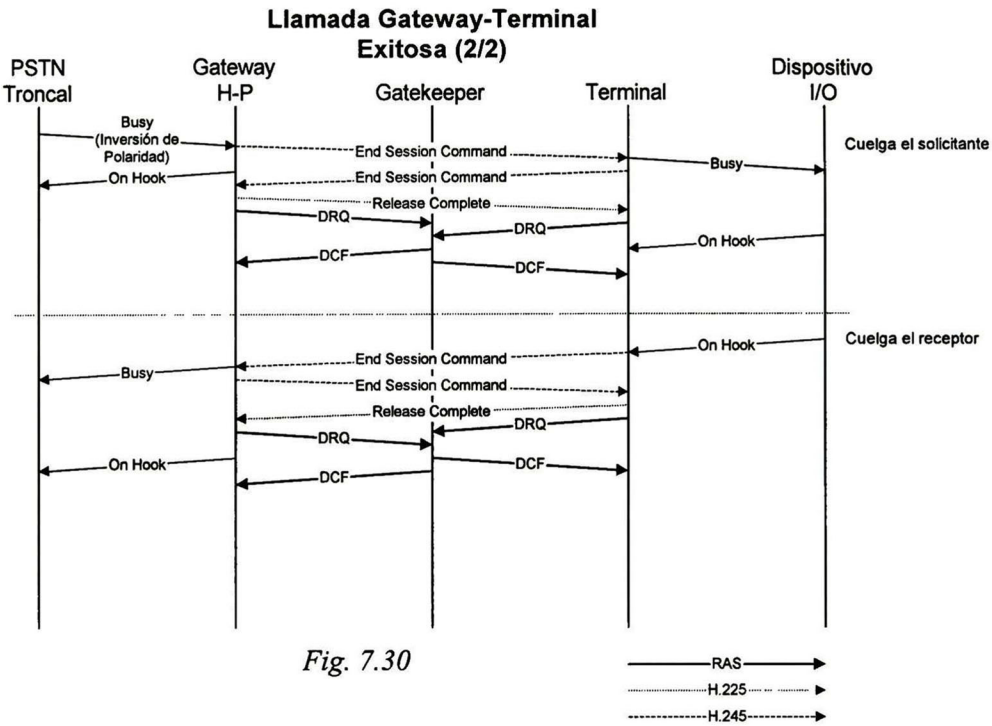


Fig. 7.30

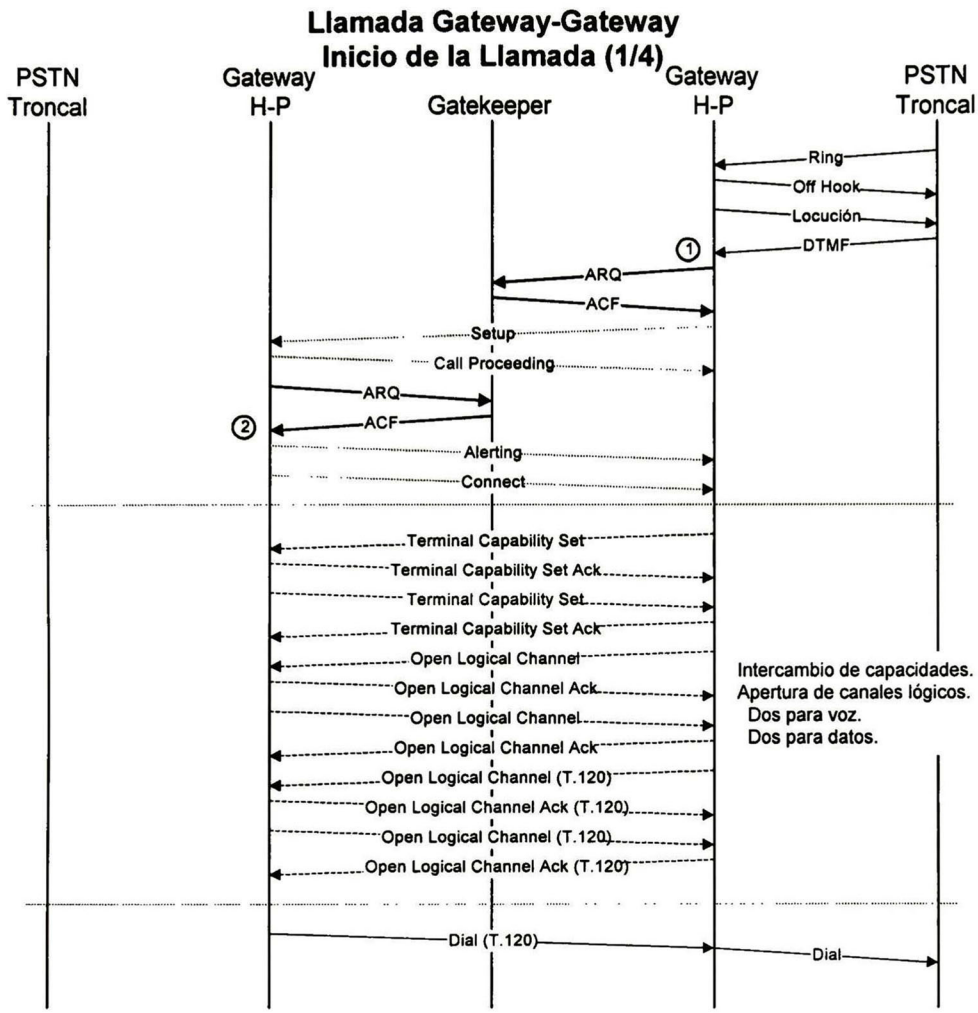
7.2.5.3.3 Cronogramas de comunicación de Gateway H-P Gateway H-P

Llamaremos a la comunicación de dos redes PSTN mediante dos Gateways H-P, ***Bypass***. Esta comunicación es una de las más atractivas de este proyecto, ya que es posible que la red sea una WAN y que esta comunicación sea transnacional por el costo de dos llamadas locales.

Las comunicaciones de red entre dos terminales de tipo Gateway H-P en una comunicación de tipo Bypass, son las mismas que en los casos anteriores, esto se debe a que estas comunicaciones están dadas por H.323 y nosotros solo nos encargamos de las comunicaciones hacia los dispositivos.

La figura 7.31 muestra el inicio de la llamada Bypass. Aquí podemos ver que un usuario de una PSTN se comunica con el otro Gateway de la misma manera que la haría con cualquier terminal.

En la figura 7.32 vemos que el usuario solicitante tiene que enviar una clave (en el caso de llamadas restringidas) al otro Gateway, de la misma manera que si una terminal estuviera deseando salir a la PSTN por un Gateway H-P. Vemos que este esquema es casi igual a la figura 7.26

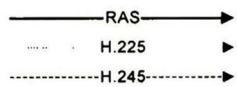


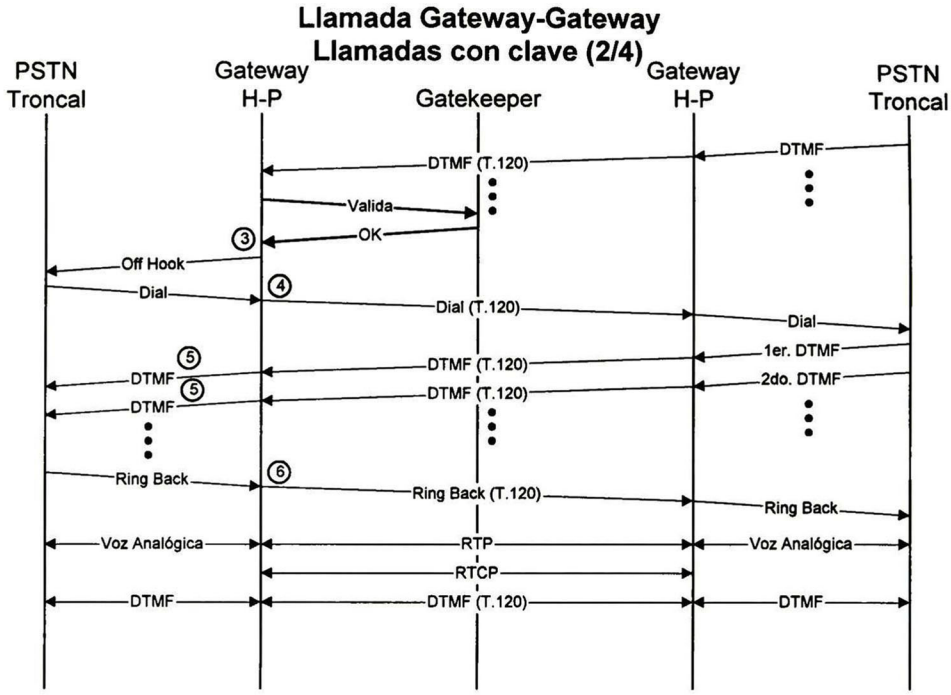
Notas:

1: Extensión del GW-HP

2: Mediante una tabla, verificar si existe línea en uso, si hay línea libre (presumiblemente) continua.
Si no, corta la llamada.
Solicita tipo de restricciones (local libre o no, prefijos restringidos). ACK (ok,0/1,00,01,...).

Fig. 7.31





Notas:

3: Recibe confirmación y restricciones asociadas a la clave. ACK(ok,00,01,...,6841580).

4: Polling hasta encontrar DIAL TONE, si no lo encuantra en ninguna de las líneas que se suponían libres, entonces corta la llamada.

5: Envía a la PSTN el dígito que le llega y verifica el dígito que envío, si concuerda con alguno de los patrones o prefijos prohibidos, corta la llamada.

6: Si recibe ocupado, corta la llamada. Si invierte la polaridad inicia tarificación.

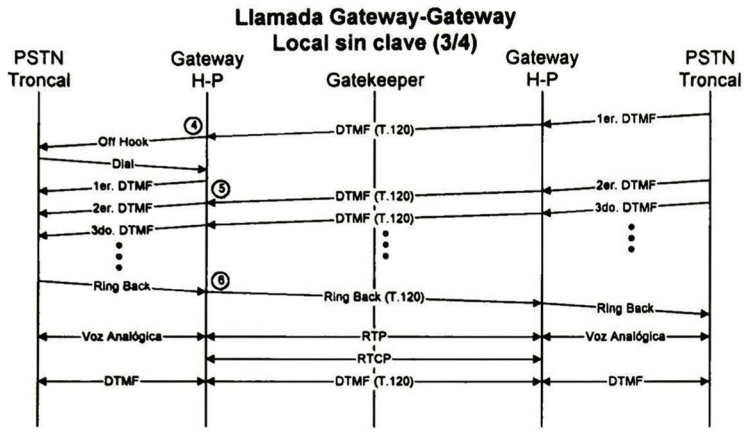
RAS
H.225
H.245

Fig. 7.32

En la figura 7.33 podemos ver lo que sucede en caso de que las llamadas locales, es decir, la salida a la PSTN, no estén restringidas, y nuevamente observamos que el diagrama es muy similar al 7.2.19.

La figura 7.34 muestra la secuencia de mensajes necesarios para realizar una desconexión en una comunicación Bypass.

También podemos observar que, nuevamente, es el Gateway que realiza la salida a la PSTN el que hace la tarificación y que para esto detecta la inversión del voltaje de la troncal en uso.



Notas.

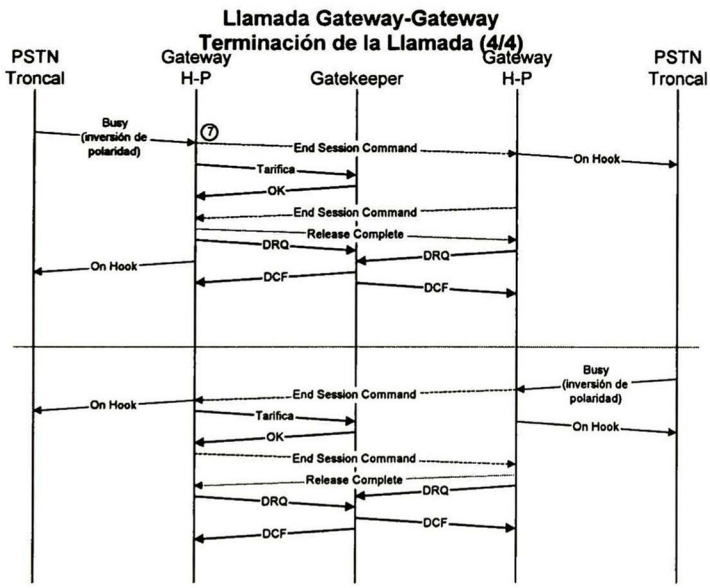
4: Polling hasta encontrar DIAL TONE, si no lo encuantra en ninguna de las líneas que se suponían libres, entonces corta la llamada.

5: Envía a la PSTN el dígito que le llega y verifica el dígito que envío, si concuerda con alguno de los patrones o prefijos prohibidos, corta la llamada.

6: Si recibe ocupado, corta la llamada. Si invierte la polaridad inicia tarificación.

RAS
H.225
H.245

Fig. 7.33



Notas:

7: Al recibir la inversión de polaridad (la llamada termina), tarifica y cierra canales

RAS
H.225
H.245

Fig. 7.34

En las figuras 7.35, 7.36, 7.37 y 7.2.38 se muestran los casos de error en una comunicación Bypass y la secuencia necesaria de mensajes para manejar estos errores.

La figura 7.35 muestra los tres primero tipos de error. El primero es cuando el Gatekeeper no permite el enlace entre los dos Gateways. En este caso, el Gateway le envía una locución al usuario de la PSTN describiendo este error y después le envía la primera locución de bienvenida.

El tercer error es cuando el Gatekeeper no permite las llamadas de salida a la PSTN. En este caso, el primer Gateway, envía al usuario de la PSTN una locución describiendo el error y después la locución de bienvenida.

En la figura 7.36, el primer error es cuando el Gatekeeper restringe las llamadas a la PSTN con una clave y el usuario no ha digitado esta clave.

La segunda parte de la figura 7.36 muestra la secuencia de mensajes cuando la llamada ha salido por la PSTN pero se ha terminado el tiempo de ring y la PSTN envía una señal de ocupado.

La última parte de esta figura es cuando el Gatekeeper restringe la salida a la PSTN con claves y la clave digitada por el usuario de la PSTN no es correcta. Aquí podemos ver un nuevo mensaje entre el Gateway y el Gatekeeper, los mensajes *valida* y *refuse*. También podemos observar que la validación de claves se hace en el Gatekeeper y no en el Gateway.

En la figura 7.37 vemos el primer error que es un intento de violación de la seguridad. El Gatekeeper ha generado una lista de restricciones sobre las llamadas hacia la PSTN (y probablemente sobre las claves de acceso) y el usuario de la primera PSTN digita un número que es restringido. En este caso, el segundo Gateway inicia el proceso de desconexión, el primer Gateway envía una locución explicando el error y después la locución de bienvenida.

El segundo error mostrado en la figura 7.37 muestra la secuencia de mensajes para manejar la falta de troncales de salida a la senda PSTN.

La figura 7.38 muestra la secuencia de mensajes necesaria para manejar un fallo en la red de datos durante una comunicación.

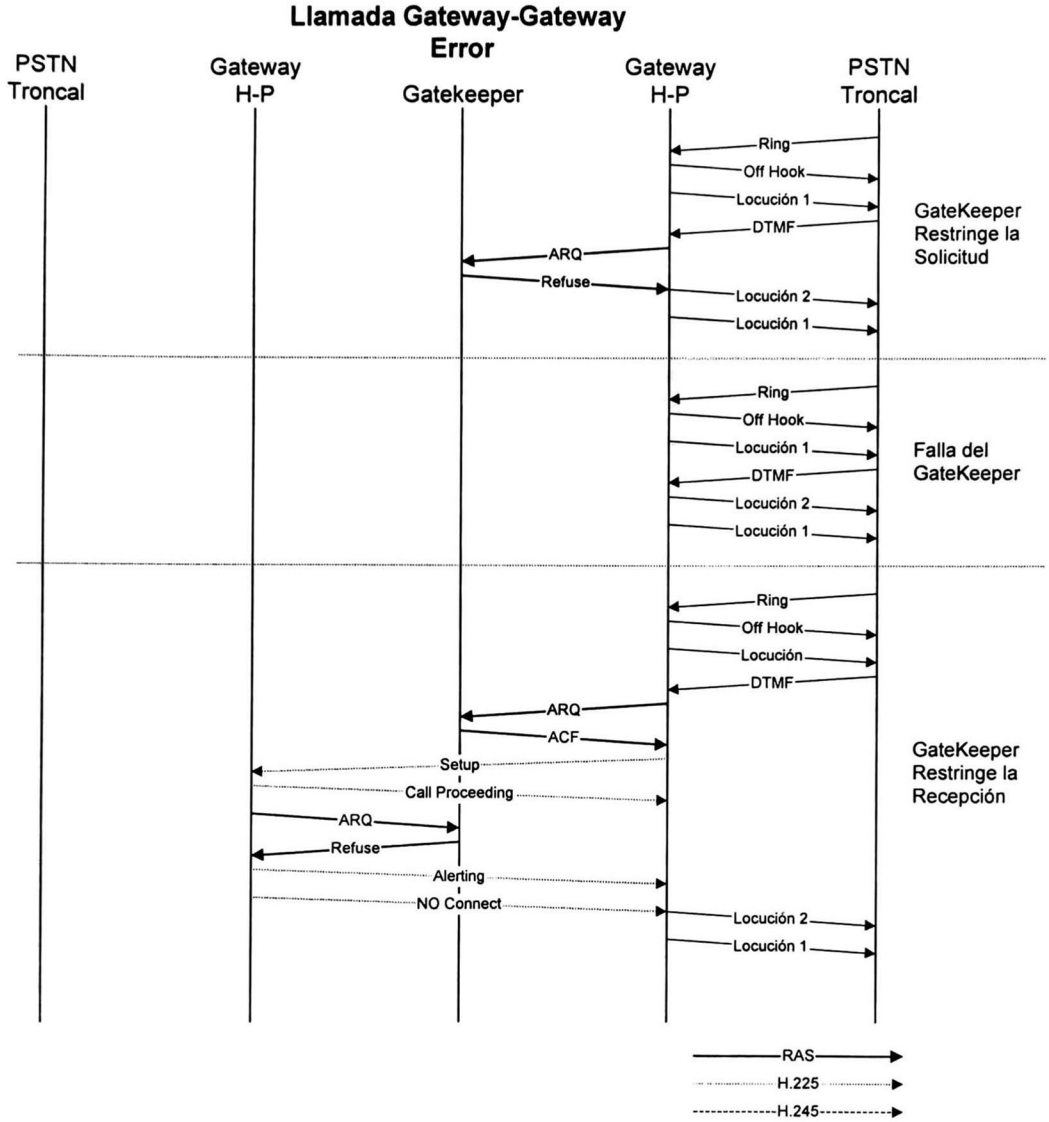

Llamada Gateway-Gateway
Error
PSTN Troncal
Gateway H-P
Gatekeeper
Gateway H-P
PSTN Troncal
Ring
Off Hook
Locución 1
DTMF
ARQ
Refuse
Locución 2
Locución 1
GateKeeper Restringe la Solicitud
Ring
Off Hook
Locución 1
DTMF
Locución 2
Locución 1
Falla del GateKeeper
Ring
Off Hook
Locución
DTMF
ARQ
ACF
Setup
Call Proceeding
ARQ
Refuse
Alerting
NO Connect
Locución 2
Locución 1
GateKeeper Restringe la Recepción
RAS
H.225
H.245

Fig. 7.35

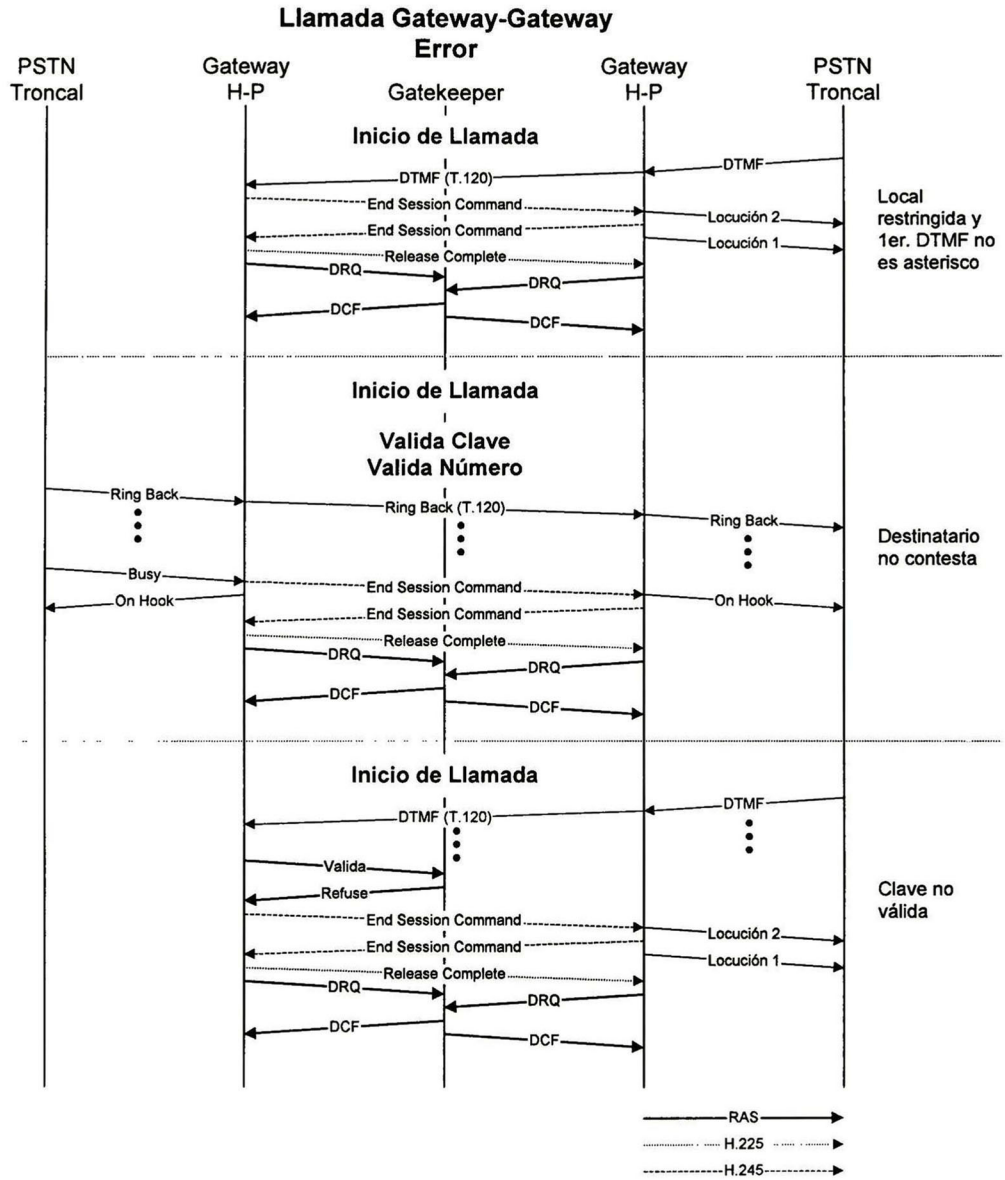

Llamada Gateway-Gateway
Error
PSTN Troncal
Gateway H-P
Gatekeeper
Gateway H-P
PSTN Troncal
Inicio de Llamada
DTMF
DTMF (T.120)
End Session Command
Locución 2
End Session Command
Locución 1
Release Complete
DRQ
DRQ
DCF
DCF
Local restringida y 1er. DTMF no es asterisco
Inicio de Llamada
Valida Clave
Valida Número
Ring Back
Ring Back (T.120)
Ring Back
Destinatario no contesta
Busy
End Session Command
On Hook
On Hook
End Session Command
Release Complete
DRQ
DRQ
DCF
DCF
Inicio de Llamada
DTMF
DTMF (T.120)
Valida
Refuse
Clave no válida
End Session Command
Locución 2
End Session Command
Locución 1
Release Complete
DRQ
DRQ
DCF
DCF
RAS
H.225
H.245

Fig. 7.36

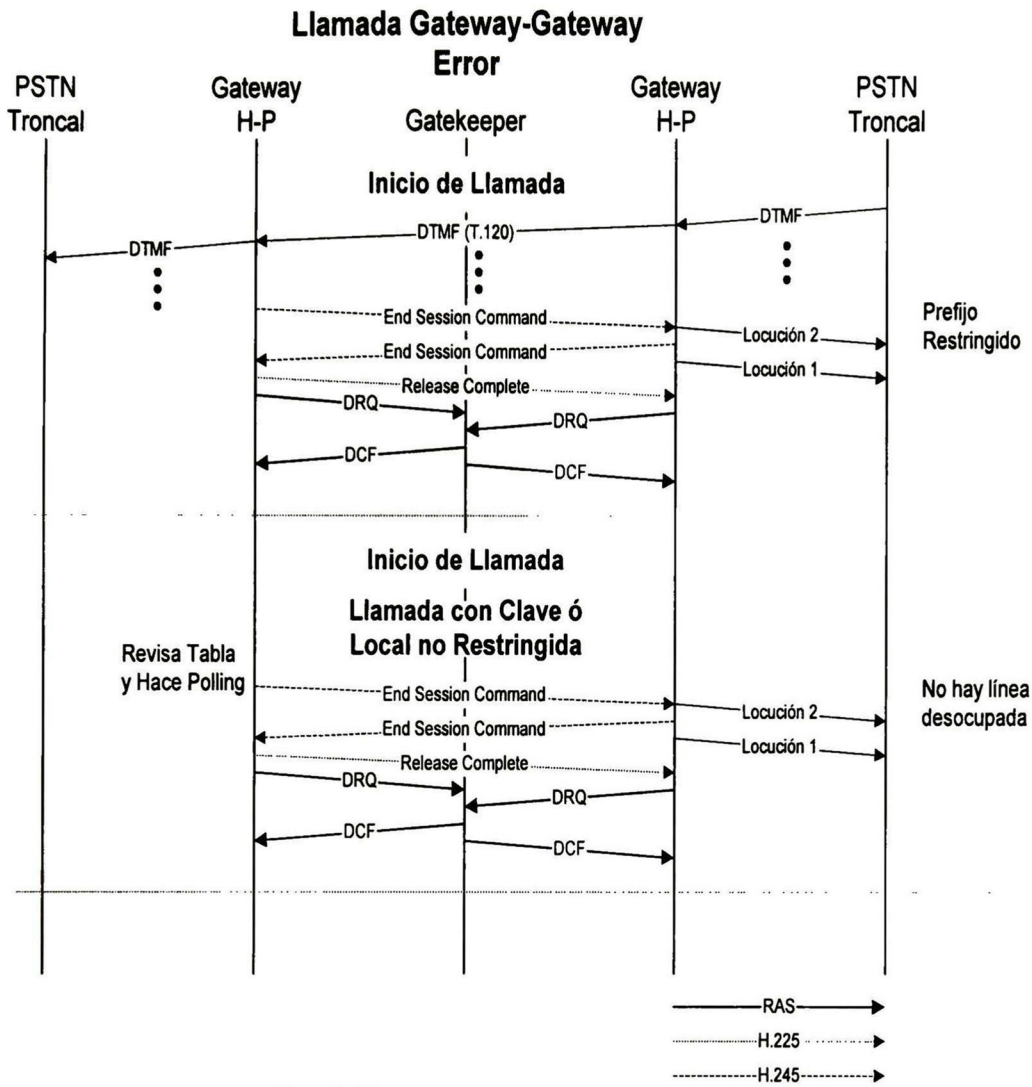

Llamada Gateway-Gateway
Error
PSTN
Troncal
Gateway
H-P
Gatekeeper
Gateway
H-P
PSTN
Troncal
Inicio de Llamada
DTMF
DTMF (T.120)
DTMF
DTMF
End Session Command
Locución 2
Prefijo
Restringido
End Session Command
Locución 1
Release Complete
DRQ
DRQ
DCF
DCF
Inicio de Llamada
Llamada con Clave ó
Local no Restringida
Revisa Tabla
y Hace Polling
End Session Command
Locución 2
No hay línea
desocupada
End Session Command
Locución 1
Release Complete
DRQ
DRQ
DCF
DCF
RAS
H.225
H.245

Fig. 7.37

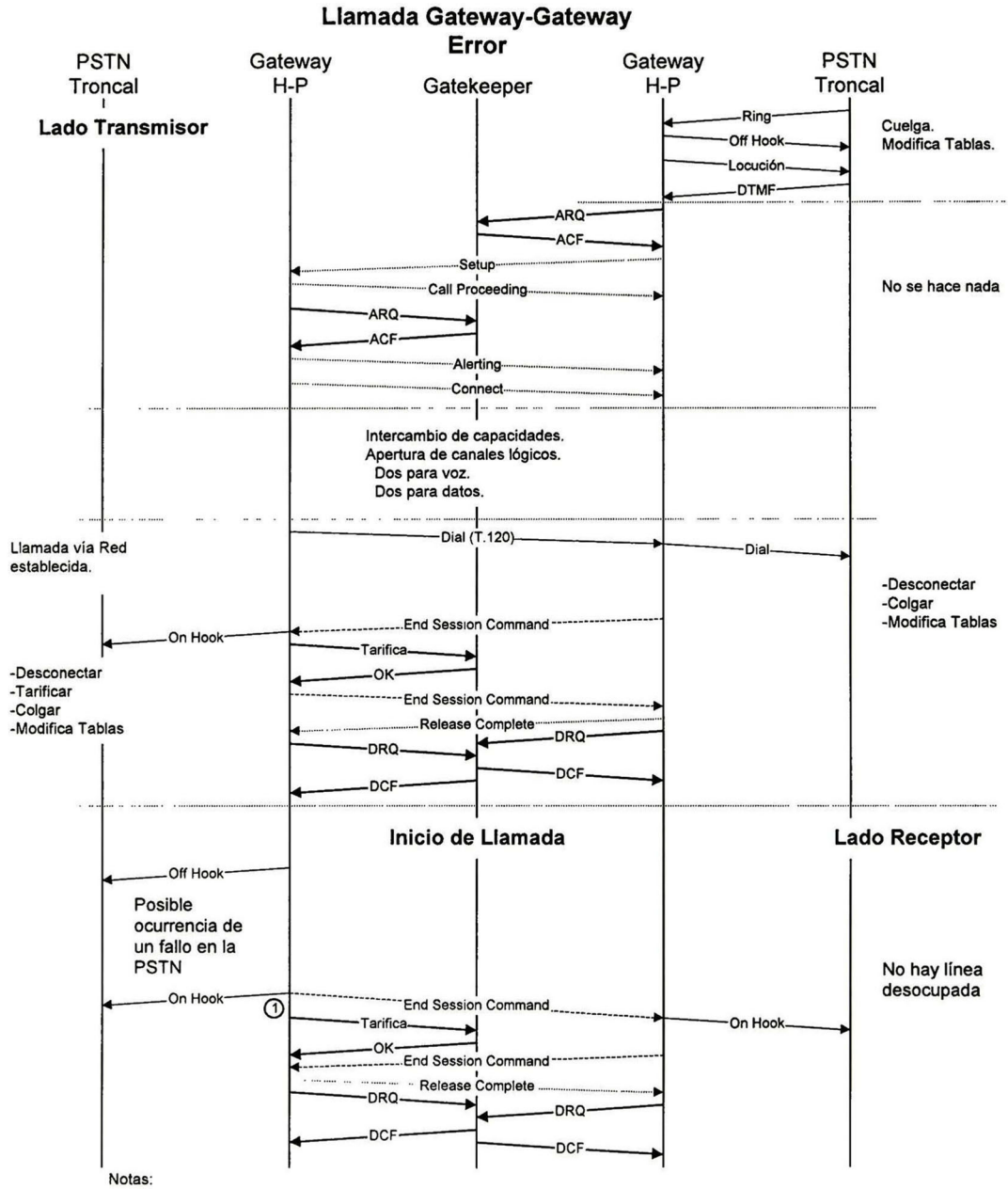

Llamada Gateway-Gateway
Error
PSTN Troncal
Gateway H-P
Gatekeeper
Gateway H-P
PSTN Troncal
Lado Transmisor
Ring
Off Hook
Locución
DTMF
Cuelga.
Modifica Tablas.
ARQ
ACF
Setup
Call Proceeding
No se hace nada
ARQ
ACF
Alerting
Connect
Intercambio de capacidades.
Apertura de canales lógicos.
Dos para voz.
Dos para datos.
Dial (T.120)
Dial
Llamada vía Red establecida.
-Desconectar
-Colgar
-Modifica Tablas
End Session Command
On Hook
Tarifica
OK
-Desconectar
-Tarificar
-Colgar
-Modifica Tablas
End Session Command
Release Complete
DRQ
DRQ
DCF
DCF
Inicio de Llamada
Lado Receptor
Off Hook
Posible ocurrencia de un fallo en la PSTN
No hay línea desocupada
On Hook
End Session Command
On Hook
Tarifica
OK
End Session Command
Release Complete
DRQ
DRQ
DCF
DCF
Notas:
1: Existe un proceso que se encarga de asegurar que las tarificaciones de llamadas llegen al GateKeeper. Se le envía un mensaje cada vez que hay que tarificar. Si no puede comunicarse con el GateKeeper, reintenta el envío hasta que sea recibido.
RAS
H.225
H.245

Fig. 7.38

Las figuras 7.39 a 7.43 muestran a detalle los mensajes entre elementos de software en una terminal suponiendo que los mensajes de red están correctos. Estos mensajes de red son representados pon una línea gruesa etiquetada como **LAN**. Los elementos involucrados son los mismos que se muestran en la figura 7.22.

Las figuras 7.39 y 7.40 muestran los mensajes en el caso de una terminal común (suponemos que esta es una PC con una tarjeta FXS como la descrita en el capítulo 4) en la que un dispositivo I/O es el teléfono conectado a la PC, por tanto los mensajes que debe manejar son los mismos que un teléfono convencional.

Las figuras 7.41, 7.42 y 7.43 muestran este mismo detalle para una comunicación entre dos Gateways H-P en la que el dispositivo de I/O se ha convertido en una troncal de PSTN.

En estas figuras podemos encontrar mensajes como: CID (Caller ID), power-up, power-down, DTMF-ini, E.S.C (End Session Command) e inversión de polaridad. Es en estos diagramas donde podemos ver el momento en que se hace la supresión de voz, el envío de CID y la recepción de inversión de polaridad.

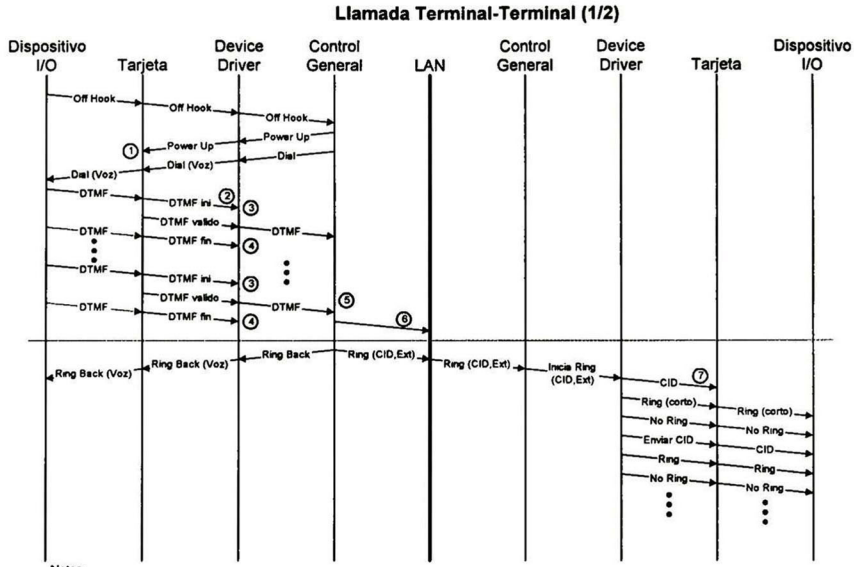
7.2.6 Consideraciones sobre la arquitectura propuesta

Debe tomarse en cuanta que el sistema fue diseñado para interactuar con módulos ya establecidos, y no desarrollados por nosotros, que implementan los protocolos necesarios de la ITU-T (h.245, h.225, RAS, T.120 y los codificadores de los medios).

En la figura 7.21 se muestra la organización del sistema en el caso de varias sesiones con un mismo control, pero hemos considerado que, ya que estos módulos están separados de la aplicación (fig. 7.12, 7.13 y 7.18), es posible que haya varias aplicaciones en la misma terminal, cada una con una instancia del bloque completo de la figura 7.21, es decir, en un momento dado, podríamos establecer varias comunicaciones, no como sesiones con el mismo control, sino como aplicaciones independientes.

Otra consideración es que estamos ocupándonos solamente de la arquitectura de software de una terminal, sin preocuparnos de un Gatekeeper (un Gateway puede ser tomado como una terminal) y de un MCU. Es en la terminal donde estamos implementando la sincronización y no en los servidores, debido a esto, podemos utilizar alguno de los Gatekeepers y MCUs comerciales.

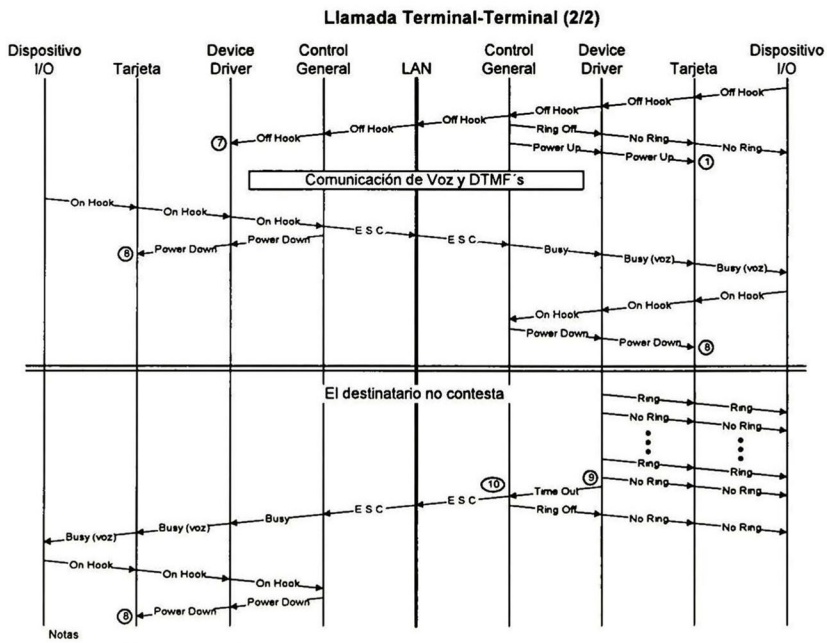
La arquitectura aquí propuesta en las secciones 7.2.1, 7.2.2, 7.2.3 y 7.2.4 debe ser compatible con los cronogramas mostrados en la sección 7.2.5. En el capítulo 8 se mostrarán las pruebas hechas a esta arquitectura.



Notas:

1: Habilita CODEC y comenzará a transmitir voz.
2: Quitar *Dial*.
3: *Device Driver* suprime la voz, trasmite silencio.
4: Transmite nuevamente voz.
5: Control General lleva la cuenta de DTMF's marcados.
6: Inicia comunicación en red.
7: Escribe en registros.

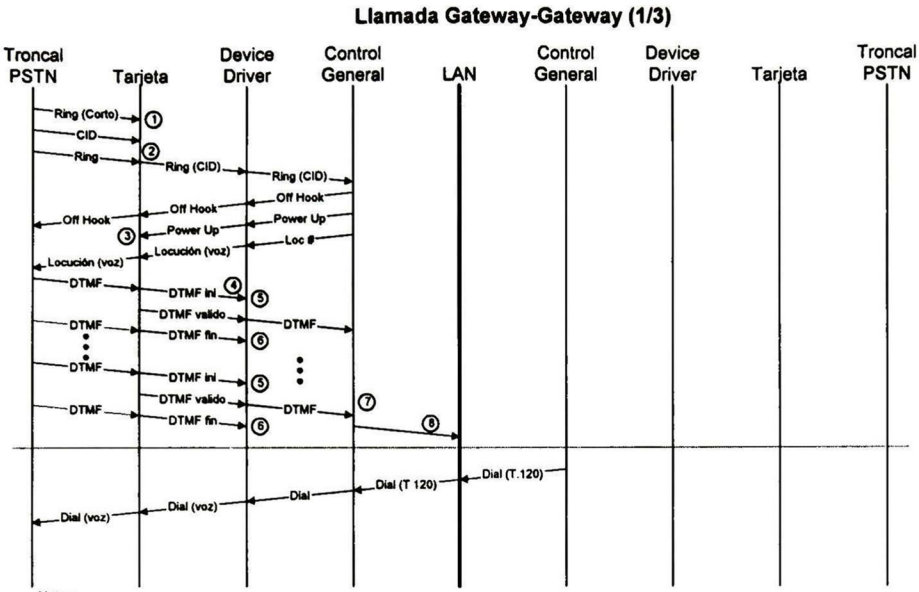
Fig. 7.39



Notas

1 Habilita CODEC y comenzará a transmitir voz.
7: Quitar *Ring Back*
8. Desactivar CODEC.
9: Device Driver lleva el conteo de rings.
10: Hace rutina de desconexión.

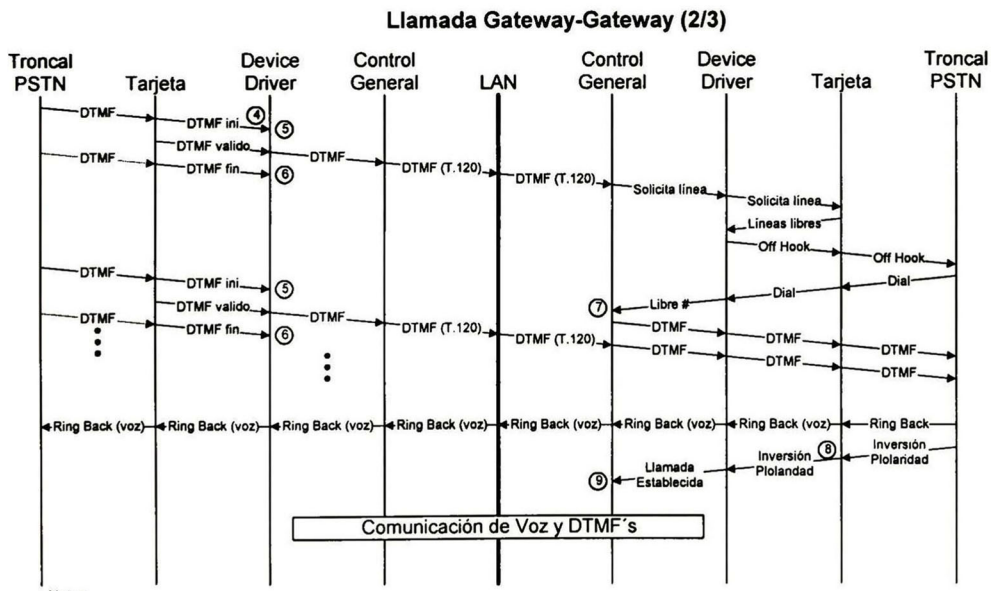
Fig. 7.40



Notas:

1 Quitar del modo de bajo consumo al Caller ID.
2: Poner en modo de bajo consumo al Caller ID.
3: Habilitar CODEC y empezar a transmitir voz.
4: Quitar *Dial* o locución.
5: Device Driver suprime la voz, transmite silencio.
6: Transmite nuevamente la voz.
7: Control General lleva la cuenta de los DTMF's marcados.
8: Inicia la comunicación conla red.

Fig. 7.41



Notas:

4: Quitar *Dial*.
5: Device Driver suprime la voz, transmite silencio.
6: Transmite nuevamente la voz.
7: Control verifica para prefijos restringidos y se marca la línea como ocupada.
8: Detecta inversión de polaridad.
9: Comienza la tarificación.

Fig. 7.42

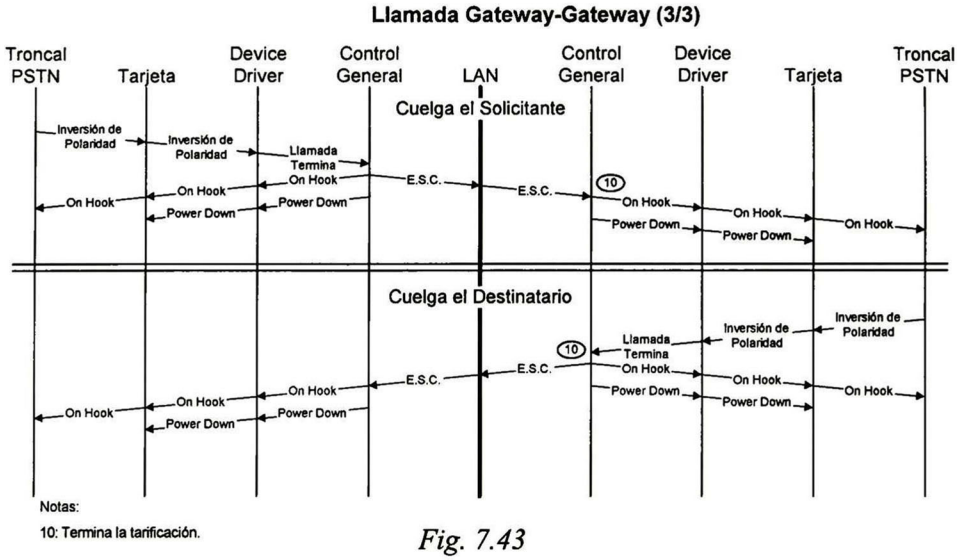

Llamada Gateway-Gateway (3/3)
Troncal PSTN
Tarjeta
Device Driver
Control General
LAN
Control General
Device Driver
Tarjeta
Troncal PSTN
Cuelga el Solicitante
Inversión de Polaridad
Inversión de Polaridad
Llamada Termina
On Hook
E.S.C.
E.S.C.
10
On Hook
On Hook
On Hook
On Hook
Power Down
Power Down
Power Down
Power Down
Cuelga el Destinatario
Inversión de Polaridad
Inversión de Polaridad
Llamada Termina
10
On Hook
E.S.C.
E.S.C.
On Hook
On Hook
On Hook
On Hook
Power Down
Power Down
Power Down
Power Down
Notas:
10: Termina la tarificación.

Fig. 7.43

8

Modelo, análisis y formalización de la arquitectura propuesta

8.1 Introducción

En este capítulo mostraremos la aplicación del método LAPEM a la simulación de la arquitectura propuesta en el capítulo 7. Mostraremos primero el modelo a bloques diseñado para la simulación, no es igual al mostrado en el capítulo 4, debido a las diferencias entre LOTOS y un lenguaje de programación, por ejemplo C++. Ya que LOTOS limita un poco la creación y destrucción de módulos en tiempo de ejecución, se ha modificado un poco la arquitectura propuesta, pero mantiene sus principales características. En la sección 8.2.2 mostraremos el listado de mensajes que pasan a través de los módulos de esta arquitectura, estos mensajes serán utilizados en la sección 8.2.3 para crear cronogramas que muestren el comportamiento del sistema. Los autómatas de comportamiento para cada módulo son mostrados en la sección 8.2.4, mientras que las secciones 8.2.5 y 8.2.6 muestran la codificación y resultados de simulación. La sección 8.3 muestra algunos comentarios sobre el capítulo, estos pueden ayudar a aclarar algunos puntos que pudieran ser confusos.

8.2 Proceso en LAPEM

Como se describió en el capítulo anterior, este proceso comienza con la representación del sistema mediante un diagrama de módulos y la obtención de sus mensajes. Después es necesario modelar el comportamiento de cada módulo mediante Máquinas de Estados Finitos para poder entonces codificar y simular el proceso completo mediante LOTOS.

8.2.1 Modelo de bloques

En la figura 8.1 podemos ver la arquitectura que será traducida a LOTOS. Esta arquitectura no es igual a la propuesta en el capítulo 7, pero mantiene las mismas características. El modelo aquí mostrado, representa una comunicación entre dos terminales multimedia, donde los mensajes de control (inicio y fin) de la llamada, son proporcionadas por un usuario a través de una interfaz. Pero estos mensajes pudieran provenir del mismo dispositivo y ser cambiados por *on-hook, off-hook*, etc. para hacerlos compatibles con los cronogramas mostrados en el capítulo 7. El modelo mostrado en este capítulo es solo uno de los casos en que la arquitectura propuesta funcionaría.

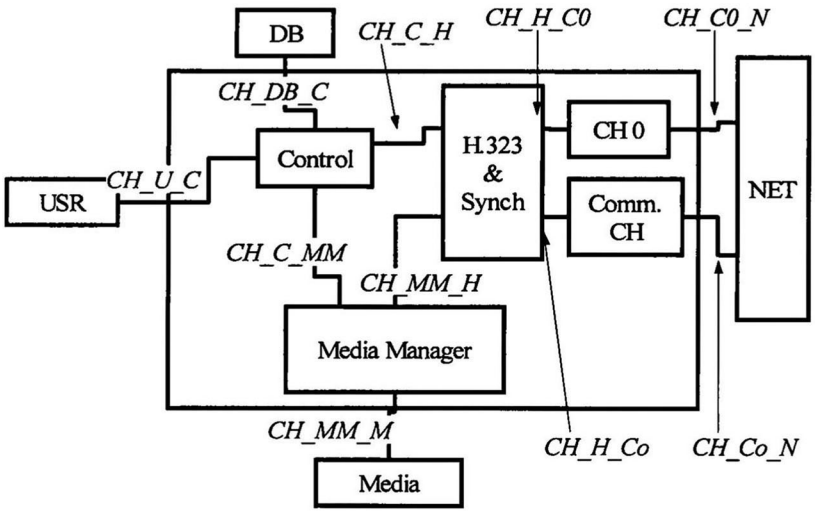


Fig. 8.1
Arquitectura para simulación

8.2.2 Listado de mensajes

A continuación un listado de los mensajes que intercambiarían los módulos mostrados en la figura 8.1. En este listado, los mensajes están agrupados por canal y dirección, de tal manera que los mensajes que se encuentran en el bloque ***módulo1 — módulo2***, y los que se encuentran en el bloque ***módulo2 — módulo1***, viajan por el mismo canal pero en direcciones contrarias.

Tabla 8.1

Mensajes	
USR — Control	**Control — USR**
Open_connection(alias)	Connection_opened
Close_connection	Connection_closed
Connection_accepted	Connection_req(alias)
Connection_rejected	

DB — Control	Control — DB
Response(alias)	Alias_to_IP(alias)
Response(IP)	IP_to_alias(IP)

Control — H.323	H.323 — Control
Init	Connection_req(IP)
Start	Connection_accepted
Stop	Connection_rejected
Close_connection	Connection_ready
Connection_req(IP)	Close_connection
Connection_accepted	
Connection_rejected	

H.323 — Channel 0	Channel 0 — H.323
Connection_req(IP)	Connection_req(IP)
Connection_accepted	Connection_accepted
Connection_rejected	Connection_rejected
C_C_exchange()	C_C_exchange()
Close_connection	Close_connection

Control — Media Man.	Media Man. — Control
Start	
Stop	

H.323 — Comm	Comm — H.323
Media_unit	Media_unit
Start	
Stop	

Comm — Net	Net — Comm
Media_unit	Media_unit

Media Man. — H.323	H.323 — Media Man.
Media_unit	Media_unit

Media Man. — Media	Media — Media Man.
Media_unit	Media_unit

Channel 0 — Net	Net — Channel 0
Connection_req(IP)	Connection_req(IP)
C_C_exchange	C_C_exchange
Close_connection	Close_connection
Connection_accepted	Connection_accepted
Connection_rejected	Connection_rejected

8.2.3 Cronogramas

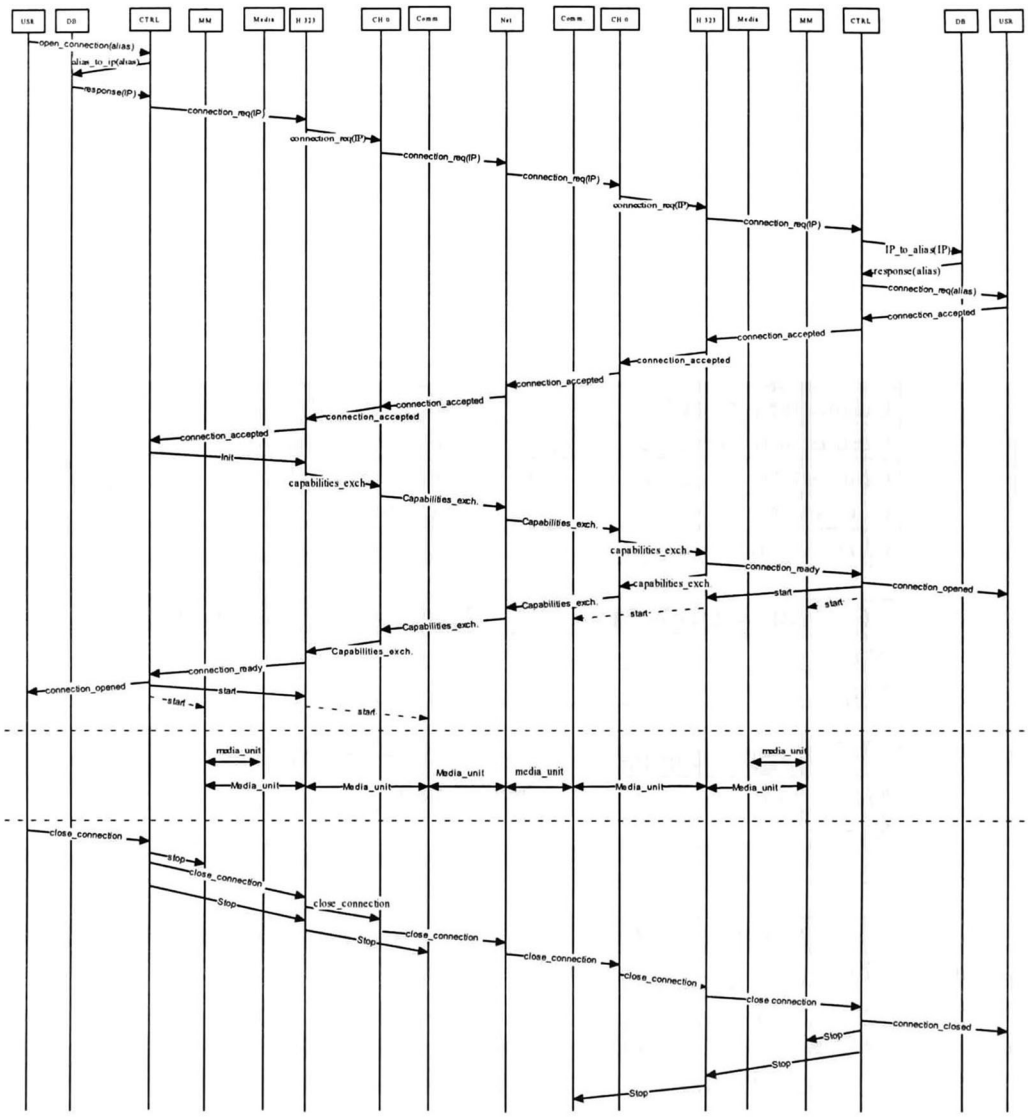
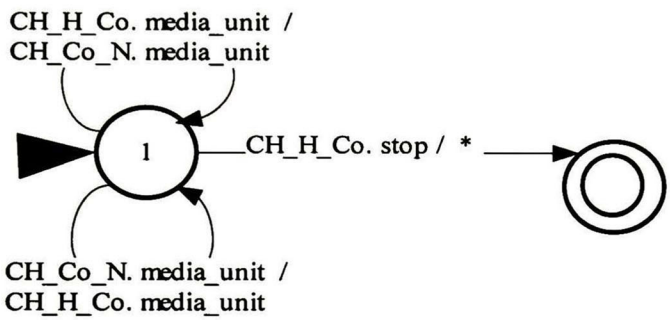


Fig. 8.2
Cronograma de comunicciones

La figura 8.2 muestra los mensajes de la tabla 8.1 en un orden cronológico que es más sencillo de analizar. En esta figura podemos ver que la primera parte, describe el proceso de conexión entre dos terminales, podemos ver que el usuario es quien da inicio al proceso de conexión y que es el usuario de la otra terminal quien decide rechazar o no la comunicación. Podemos ver también que es el módulo etiquetado como *H.323 & Sync.* el encargado de hacer el intercambio de capacidades (proceso propio del estándar H.323 de la

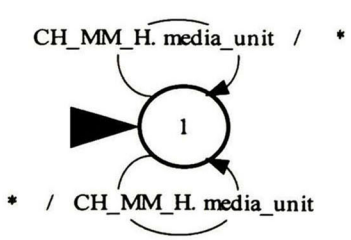
ITU – T) y es quien maneja el canal principal de comunicación, representado por el módulo etiquetado como *canal 0*, que representa el canal 0 manejado por el estándar H.245 de la ITU en H.323. Es también éste módulo *H.323 & Sync.* el encargado de instanciar a los canales de comunicación de los medios, representados aquí por el módulo *Comm. Ch.* y que representa a los codificadores recomendados por la ITU en H.323. Vemos también que hay un mensaje de instanciación entre el módulo de control y el módulo *media manager.* Este módulo media manager, representa a los módulos *VxD, API y device management* mostrados en el capítulo 7 y puede variar dependiendo del dispositivo de medio que se esté manejando, vgr. teléfono o equipo multimedia.

8.2.4 Autómatas

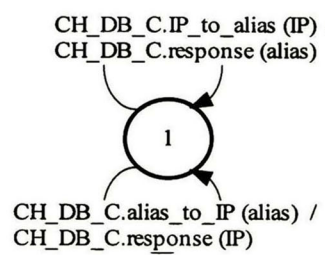


Autómata para el canal de comunicación de medios
Fig. 8.3

La figura 8.3 muestra el autómata que modela el comportamiento del canal de comunicación de los medios. Este módulo representa los canales dinámicos (RTP) que portarán a los diferentes medios. Ya que es un canal dinámico, deberá tener un estado final que representa el punto en el que el canal es destruido, en programación orientada a objetos, esto es hecho por el destructor del objeto. Como se ve en la figura, este módulo solamente transmite lo que recibe de la terminal a la red y viceversa y es destruido por el módulo etiquetado como *H.323 & Sync.*



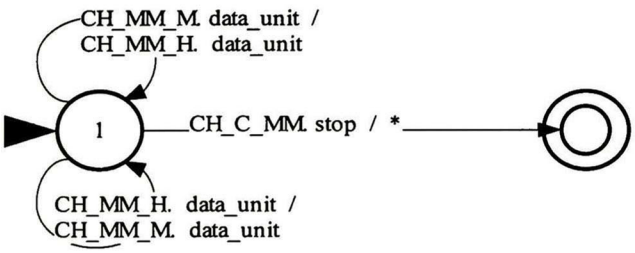
Autómata de comportamiento de Media
Fig. 8.4



Cronograma de comportamiento de la DB
Fig. 8.5

La figura 8.4 representa el comportamiento del módulo *Media* que solamente genera o reproduce unidades de medios, lo que en el capítulo 2 se llaman ULDs.

La figura 8.5 muestra el comportamiento del módulo de base de datos que traduce las direcciones IP a un registro que contiene la información correspondiente al usuario destino y que nosotros llamamos *Alias*. Este módulo realiza también el proceso inverso.

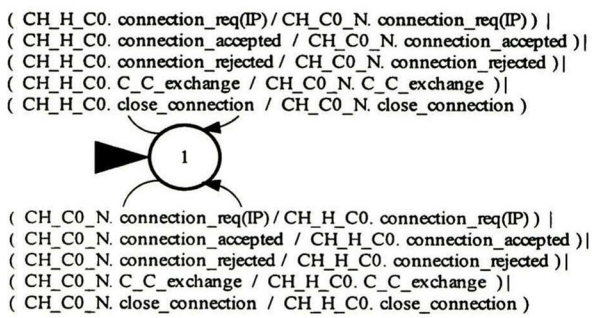


Cronograma de conpurtamiento de Media Management

Fig. 8.6

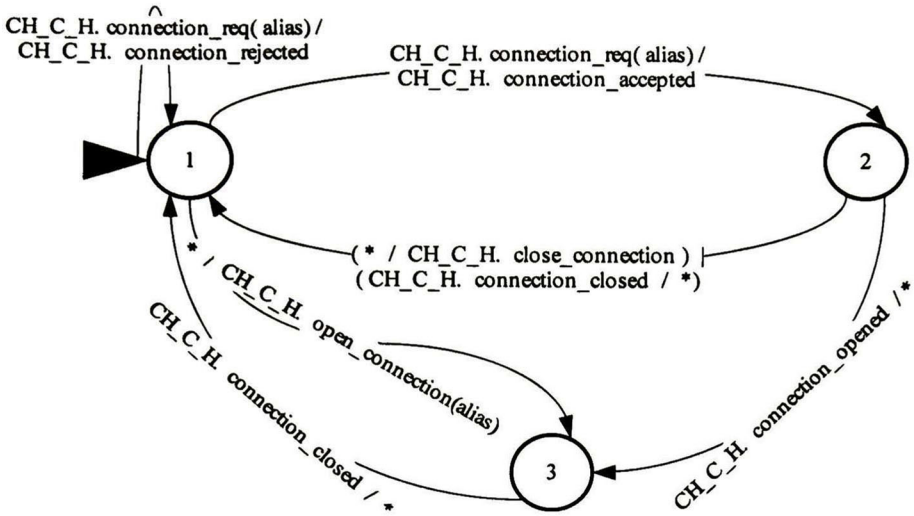
La figura 8.6 muestra el comportamiento del módulo de manejo de medios. Este módulo es dinámico y es instanciado por el módulo de *control*. Esto es representado en FSM mediante un estado final que es alcanzado con un mensaje de *control*. Este trabajo sería hecho por el método destructor en programación orientada a objetos. Este módulo representa al *device management, API y VxD* mostrados en el capítulo 7 y dependen del tipo de medio que se maneje. El comportamiento mostrado en este autómata es solamente pasar los mensajes del medio hacia el módulo etiquetado como *H.323 & Sync.* y viceversa.

La figura 8.7 muestra el funcionamiento del canal 0, este es el canal manejado por H.245 en la recomendación H.323 de la ITU. El trabajo de este canal es solo transportar los mensajes del sistema a la red y viceversa, es decir, no toma decisiones.



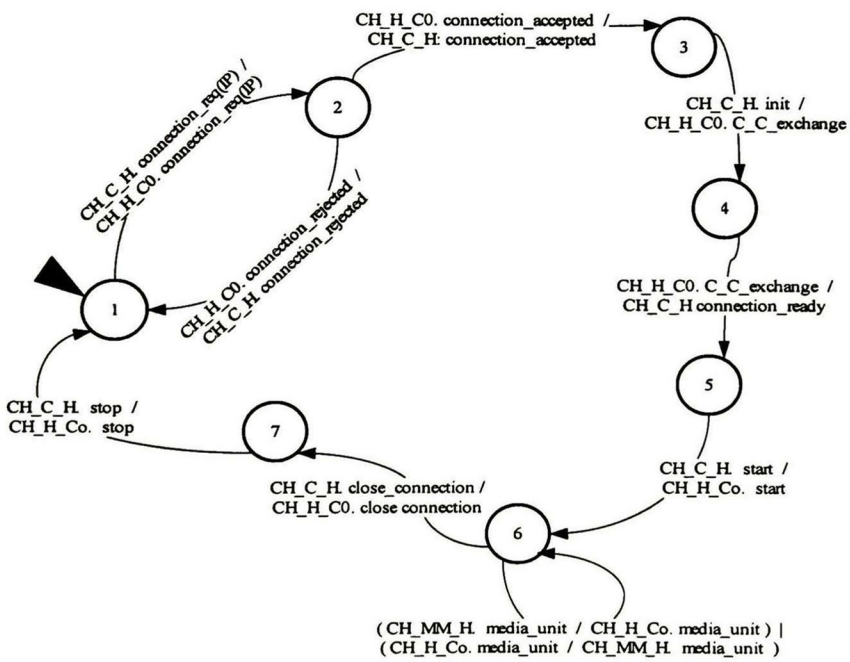
Autómata de comportamiento del Canal 0

Fig. 8.7



Autómata de comportamiento del Usuario

Fig. 8.8



Autómata del comportamiento del módulo H.323

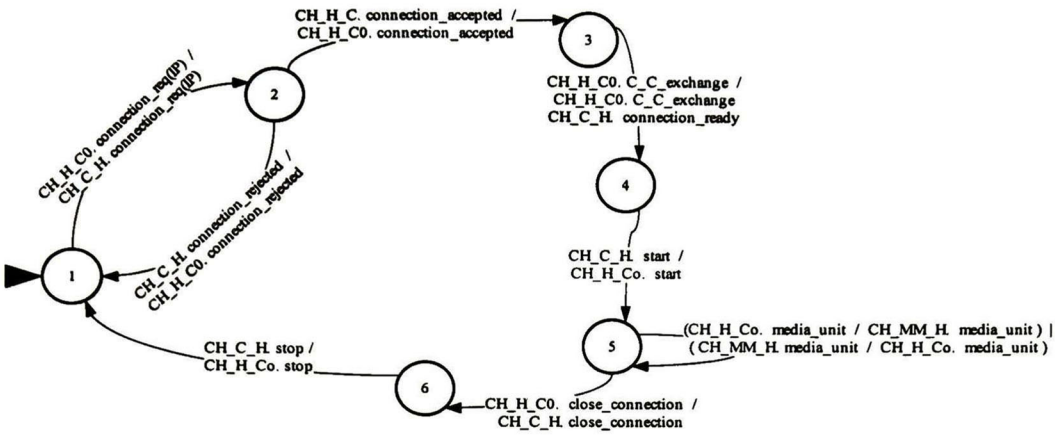
Fig. 8.9

La figura 8.8 muestra el comportamiento del Usuario. Este comportamiento es sencillo ya que solamente da instrucciones de abrir o cerrar una conexión, o de aceptar o rechazar esta misma. El resto de la información es transmitida a través del módulo *Media.*

El módulo etiquetado como *H.323 & Sync.* es modelado de distinta manera. Primero, la figura 8.9 muestra el comportamiento de este módulo en la terminal que solicita la conexión, por tanto sus primeros mensajes llegan desde el *Control.* Transmite esta solicitud por el *canal 0* y espera respuesta, en caso de que sea negada la conexión, se notifica al *Control* y se regresa al estado inicial, de otra manera se notifica al *Control* y se espera la orden para iniciar el intercambio de capacidades y características (procedimiento definido en la recomendación H.323). Después de esto, se instancian los canales de comunicación de medios (RTP, representados aquí por el módulo *Canal 0*) e inicia la transferencia de ULDs del manejador del medio a la red y viceversa. En cualquier momento en este punto, puede recibir una solicitud de desconexión del *Control*, en este caso, envía una mensaje de paro al canal de medios y un mensaje de desconexión a la red, para regresar al estado inicial.

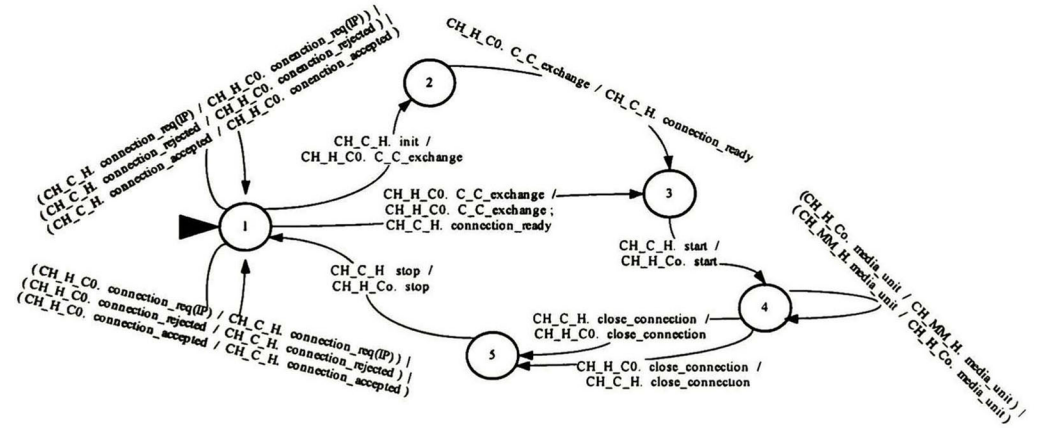
La figura 8.10 muestra el comportamiento de este mismo módulo, pero en la terminal que recibe la solicitud de conexión. Cuando recibe este mensaje, notifica al control y espera respuesta, en caso de que el *Usuario* rechace la conexión, envía un mensaje de cierre de conexión a la red y regresa al estado inicial. En otro caso, envía un mensaje de aceptación a la red y espera por el inicio de intercambio de características y capacidades. Una vez hecho este proceso, notifica al *Control* y espera el mensaje de inicio con el que instancia al módulo de comunicación de medios (RTP). Ahora este módulo se dedica solamente a transferir los mensajes del medio a la red y viceversa. En cualquier momento en este punto, puede recibir una solicitud de desconexión de la red y avisar al *Control* y enviar la señal de fin al canal de medios y regresar al estado inicial.

La figura 8.11 muestra un autómata donde se conjuntan los dos anteriores y que modela el comportamiento para el módulo *H.323 & Sync.* para los dos casos mostrados: terminal solicitante y terminal receptora. Aquí podemos ver que en estado inicial, este módulo espera una solicitud de parte del usuario de su terminal o de la red y que se mantiene ahí hasta recibir una aceptación de cualquiera de las dos partes. , entonces cambia ale estado 3 después de haber hecho el intercambio de capacidades y características. Espera entonces un mensaje de *Control* para instanciar al módulo de comunicación de medios. Y se queda en un estado de transferencia de ULDs entre *Control* y el canal de medios, del cual sale con una solicitud de desconexión de cualquiera de las dos partes, y regresa al estado inicial después de haber descargado al módulo de comunicación de medios.



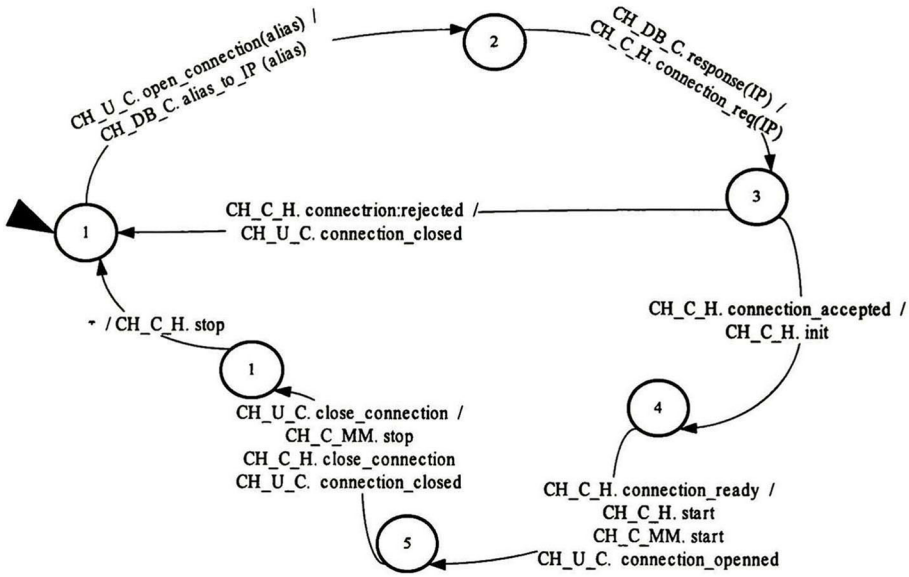
Autómata del comportamiento del módulo H.323

Fig. 8.10



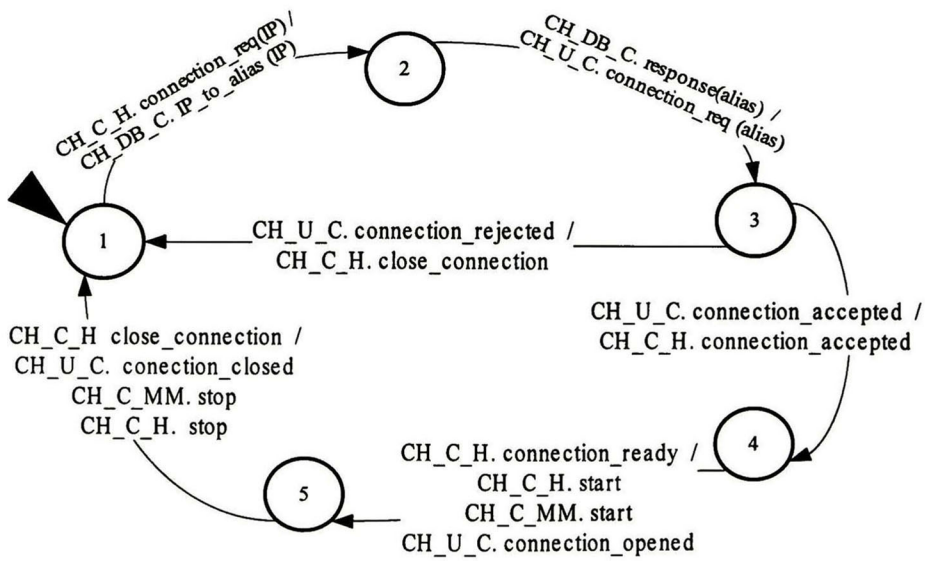
Autómata del comportamiento del módulo H.323

Fig. 8.11



Autómata de comportamiento de Control

Fig. 8.12



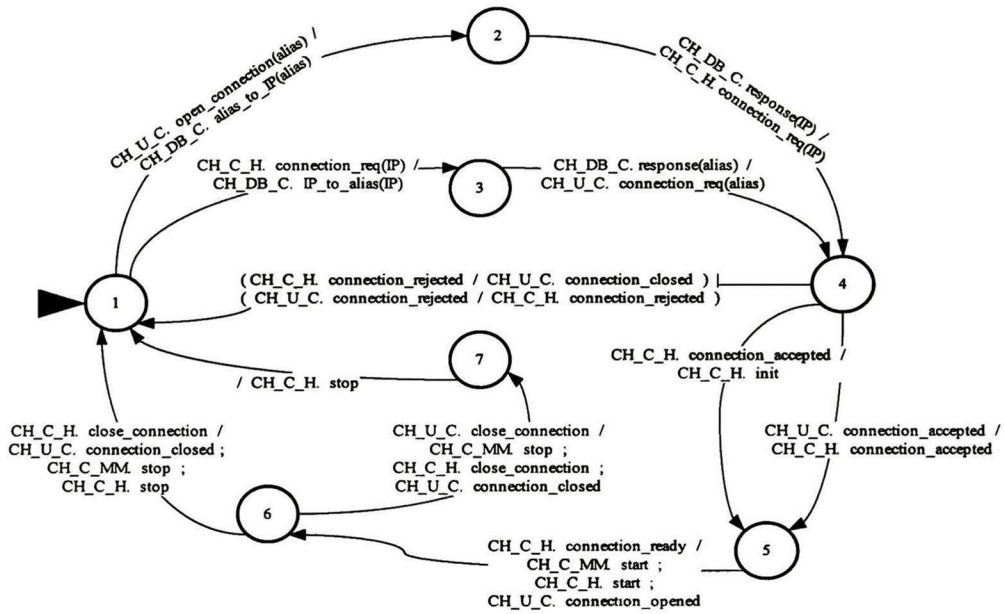
Autómata de comportamiento de Control

Fig. 8.13

En la figura 8.12 podemos observar el comportamiento del módulo de *Control* para la terminal que solicita la conexión. En este caso, la acción en este módulo es iniciada por mensajes del Usuario. Es este módulo el encargado de solicitar el cambio de *Alias* a IP y viceversa. Una vez enviado el mensaje de solicitud de conexión, espera por la respuesta. Si esta respuesta es negativa, regresa al estado inicial después de haber notificado al usuario de este evento, en otro caso, cambia a un estado de espera mientras el módulo *H.323 & Sync.* realiza el intercambio de capacidades y características. Cuando este mensaje es recibido, instancia al módulo *media manager* y cambia a un estado de espera de desconexión, esta puede ser recibida desde el usuario o desde la red.

La figura 8.13 muestra el comportamiento del módulo de *Control* para la terminal que recibe la solicitud de conexión. Aquí, permanece en el estado inicial mientras no reciba una solicitud de conexión de la red, en cuyo caso notifica al usuario para que este acepte o rechace la conexión. En caso de ser rechazada, *Control* notifica a la red y regresa al estado inicial, de otra manera, cambia a un estado de espera en que el módulo *H.323 & Sync.* realiza el intercambio de características y capacidades. Cuando es notificado del fin de esta acción, realiza una instanciación del módulo *Media Manager* y notifica al módulo *H.323 & Sync.* que instancie los canales de comunicación de medios. Este autómata regresa al estado inicial cuando recibe de la red una solicitud de desconexión, en este caso, envía un mensaje de fin a *H.323 & Sync.* y a *Media Manager.*

Ya que una terminal puede recibir una solicitud desde un usuario o desde la red, debe tener los dos comportamientos señalados en los párrafos anteriores. La figura 8.14 muestra el autómata que conjunta estos dos comportamientos. En esta figura podemos observar que este módulo se mantiene en su estado inicial hasta que reciba una solicitud de conexión de la red o de un usuario y que regresa a su estado inicial en caso de que reciba un mensaje de desconexión de cualquiera de las dos partes. En caso de recibir una respuesta afirmativa, cambia a un estado de espera en que el módulo *H.323 & Sync.* realiza el intercambio de características y capacidades, y después instancia al módulo *Media Manager*, notifica al usuario del establecimiento de la conexión y manda un mensaje al módulo *H.323 & Sync.* para que instancie al canal de comunicación de los medios. Se mantiene en este estado hasta que reciba una solicitud de desconexión del usuario de la red. En ambos casos, envía una notificación de desconexión al usuario, un mensaje de fin al módulo *H.323 & Sync.* para que destruya los canales de comunicación de medios y un mensaje de fin al módulo *Media Manager* para que finalice (o se destruya, en una implementación real).



Autómata de comportamiento de Control

Fig. 8.14

8.2.5 Codificación en LOTOS

A continuación mostramos el ejemplo de la codificación en LOTOS de los autómatas mostrados en la sección anterior.

```
(*************************************************************************)
(***                                                                   ****)
(***   Simulación del sistema de control, trasmisión y algoritmos de   ****)
(***   sincronización para el sistema VOIP                             ****)
(***                                                                   ****)
(***                                                                   ****)
(***                                                                   ****)
(***    David Garduño, Guadalajara, Jal. México.                       ****)
(***                   Diciembre 1999                                  ****)
(***                                                                   ****)
(*************************************************************************)
(*************************************************************************)
(*************************************************************************)


SPECIFICATION arquit_VoIP[CH_DB_C_1, CH_U_C_1, CH_C_MM_1, CH_MM_H_1, CH_MM_M_1,
                          CH_C_H_1, CH_H_C0_1, CH_H_Co_1, CH_C0_N_1, CH_Co_N_1,
                          CH_DB_C_2, CH_U_C_2, CH_C_MM_2, CH_MM_H_2, CH_MM_M_2,
                          CH_C_H_2, CH_H_C0_2, CH_H_Co_2, CH_C0_N_2,
                          CH_Co_N_2]:NOEXIT
```

```
(*************************************************************************)
(**     COMPOSICION                                                     **)
(*************************************************************************)

BEHAVIOR
(((
(((((Est_Ini_USR[CH_U_C_1] |[CH_U_C_1]|
EST_Ini_Control[CH_U_C_1, CH_C_MM_1, CH_DB_C_1, CH_C_H_1,
                CH_MM_M_1, CH_MM_H_1])|[CH_DB_C_1]|
Est_Ini_DB[CH_DB_C_1])|[CH_C_H_1]|
EST_Ini_H323[CH_C_H_1, CH_MM_H_1, CH_H_C0_1, CH_H_Co_1, CH_Co_N_1])|[CH_H_C0_1]|
Est_Ini_Canal_0[CH_H_C0_1, CH_C0_N_1]) |[CH_C0_N_1, CH_Co_N_1]|
Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2])
|[CH_C0_N_2, CH_Co_N_2]|
((((Est_Ini_USR[CH_U_C_2] |[CH_U_C_2]|
EST_Ini_Control[CH_U_C_2, CH_C_MM_2, CH_DB_C_2, CH_C_H_2,
                CH_MM_M_2, CH_MM_H_2])|[CH_DB_C_2]|
Est_Ini_DB[CH_DB_C_2])|[CH_C_H_2]|
EST_Ini_H323[CH_C_H_2, CH_MM_H_2, CH_H_C0_2, CH_H_Co_2, CH_Co_N_2])|[CH_H_C0_2]|
Est_Ini_Canal_0[CH_H_C0_2, CH_C0_N_2])
)|[CH_MM_M_1]|
Est_Ini_Media[CH_MM_M_1])     |[CH_MM_M_2]|
Est_Ini_Media[CH_MM_M_2])


WHERE
```

8.2.6 Resultado en LOTOS

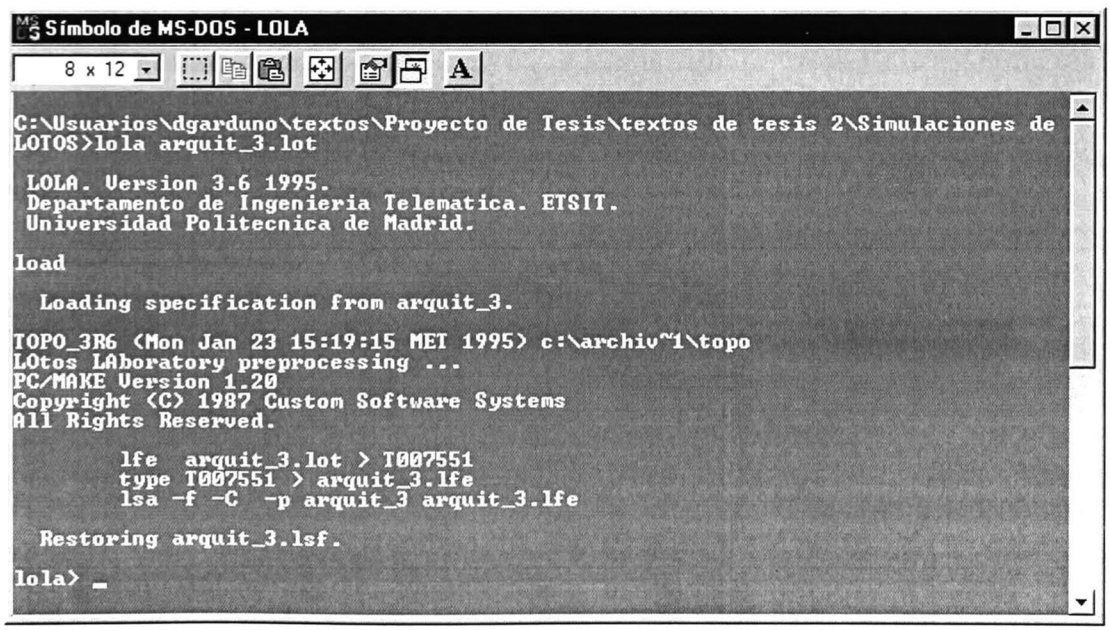


Fig. 8.15
Compilación exitosa

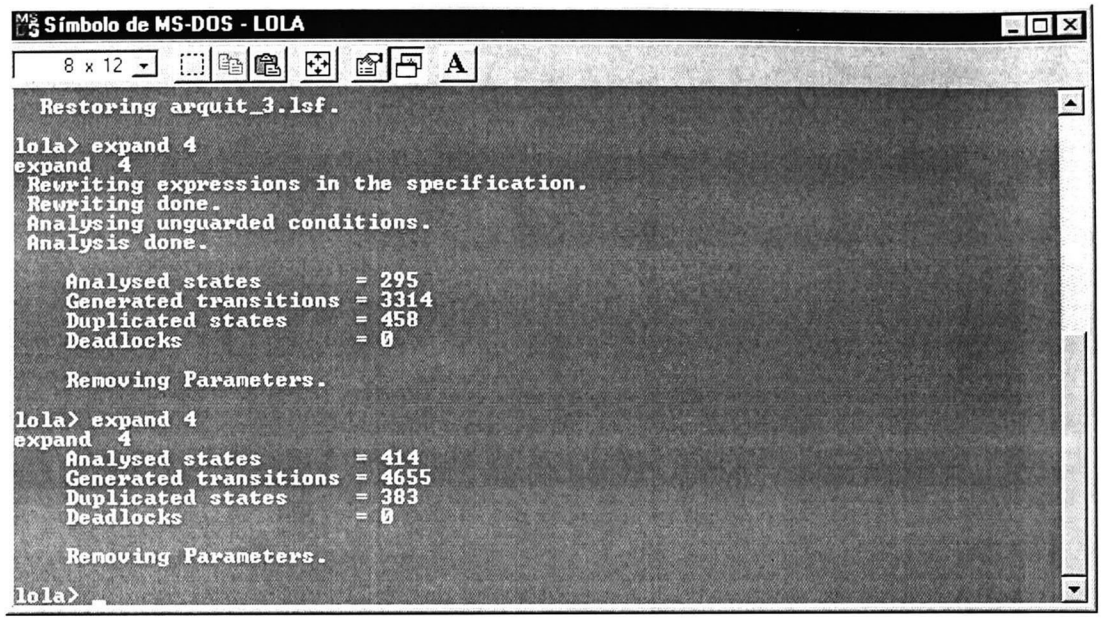


Fig. 8.16
Expansión del árbol de eventos

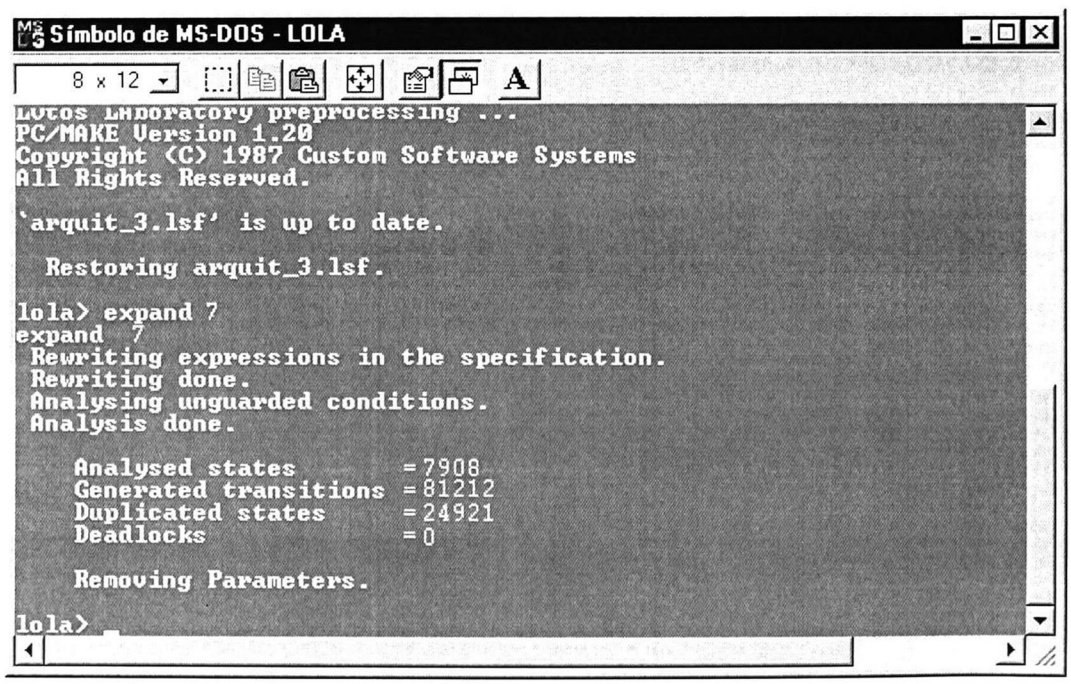


Fig. 8.17
Expansión del árbol de eventos

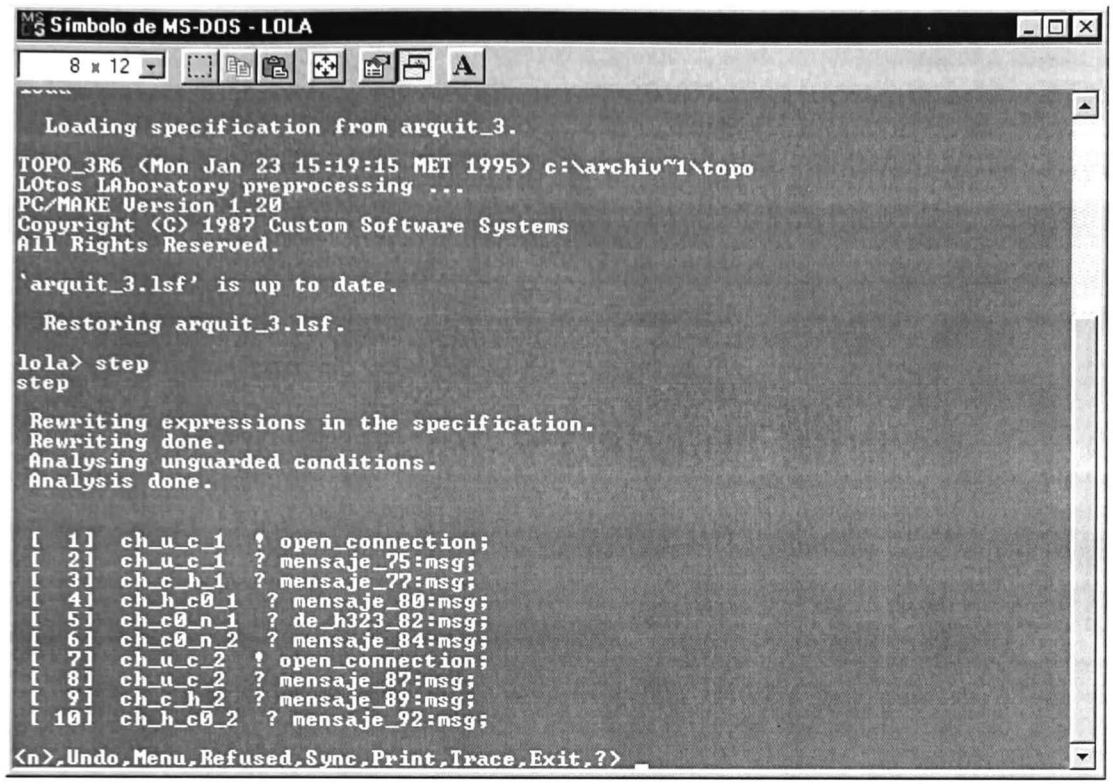


Fig. 8.18

Disposición inicial de eventos

Las figuras 8.15 a 8.18 muestran los resultados de la simulación hecha al código anterior. Estas simulaciones se hicieron utilizando un compilador de LOTOS hecha por la Universidad de Valencia, llamada LOLA (LOTOS Laboratory).

La figura 8.15 muestra la compilación exitosa del código. La figura 8.16, 8.17 y 8.18 muestran el resultado del árbol de expansión del modelo global, en ellas podemos ver que no hay deadlocks, esto quiere decir que para todos los mensajes enviados, siempre hay otro módulo que los reciba en cualquier momento.

La figura 8.18 muestra el inicio de la simulación paso a paso en la que podemos resaltar las transiciones 1 y 7 en que cualquier a de los dos usuarios pueden iniciar una conexión.

9

Conclusiones y trabajos futuros

9.1 Conclusiones

En este trabajo se ha presentado una descripción formal de requerimientos para un sistema de comunicación de voz en redes IP. También se presentó una metodología que ayuda al análisis y diseño de sistemas distribuidos. Además se ha mostrado una arquitectura de software que parte de la recomendación ITU-T H.323 y por tanto es compatible con la misma. Esta arquitectura fue modelada mediante máquinas de estados finitos y simulada con el lenguaje LOTOS.

Como primer resultado de este trabajo, está la metodología LAPEM desarrollada para el análisis y diseño de protocolos de comunicación. Esta metodología mostró ser más rápida e intuitiva que otras también descritas aquí.

Hemos obtenido una especificación de requerimientos y diseño para un sistema que integra una red de datos a una red de voz. En esta especificación se asientan los requerimientos de algunos servicios digitales telefónicos y se sientan las bases para nuevos servicios. Se muestran también las características de la entidad administradora, sus tareas y restricciones.

También se propuso, modeló y simuló una arquitectura de software que implementa un sistema de voz sobre IP y que soporta mecanismos de sincronización, cumpliendo así con el principal objetivo de este trabajo. De esta manera, se aportan nuevas facilidades a la recomendación H.323.

Hemos también podido observar que esta misma arquitectura de software podría ser fácilmente adaptada a un sistema de conmutación de voz para redes de datos. Este caso, aunque es bastante interesante, no ha sido contemplado aquí.

9.2 Trabajos futuros

Como una posible extensión del trabajo aquí presentado y en base a los resultados obtenidos, creemos factible la realización de los siguientes trabajos:

- Desarrollar una cama de pruebas para el sistema de transmisión de voz sobre una red de datos local
- Implementar la arquitectura propuesta, realizar pruebas de rendimiento y hacer un estudio comparativo con las redes telefónicas convencionales.
- Desarrollar una cama de pruebas para el sistema de transmisión de voz, audio, video y datos sobre una red de datos local
- Hacer una ampliación de las camas de prueba anteriores para su uso en redes WAN
- La construcción de mejores herramientas para la validación de sistemas distribuidos

Apéndice A: Ejemplo de aplicación de LAPEM

Problema: Se desea especificar, analizar y diseñar un protocolo de comunicaciones como FTP, pero simplificándolo para fines demostrativos.
Las características de este protocolo, que llamaremos SFTP (Simplified FTP), se muestran a continuación:

9. El usuario, mediante el *Cliente*, se conecta con el *Servidor*.
10. Para ingresar al servidor, se requiere de un *ID* y un *Password*.
11. El usuario puede ejecutar comandos locales.
12. El usuario puede ejecutar comandos remotos.
13. El usuario puede ejecutar comandos de transferencia, esto es, puede hacer una transferencia de archivos desde el servidor al local.
14. El usuario puede salir.
15. En caso de iniciar una conexión y que el servidor rechace tal solicitud, el usuario deberá iniciar en el paso 1.
16. La ejecución de cualquiera de los comandos por el usuario, generará un aviso o notificación del resultado de la ejecución.

Paso 1. Se hace un diagrama a bloques del sistema visto desde la perspectiva del usuario.

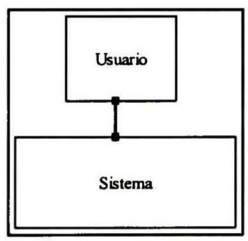


Fig. A.1

Paso 2. Refinamiento

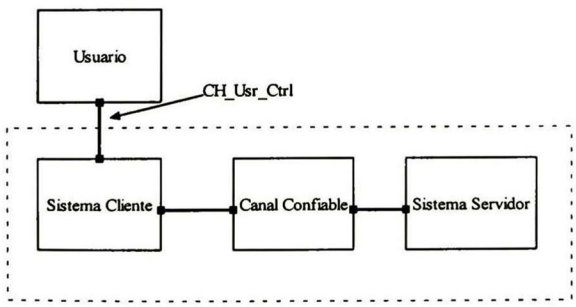


Fig. A.2

Paso 2.1. Refinamiento, se convierten las cajas negras a cajas blancas de manera iterativa.

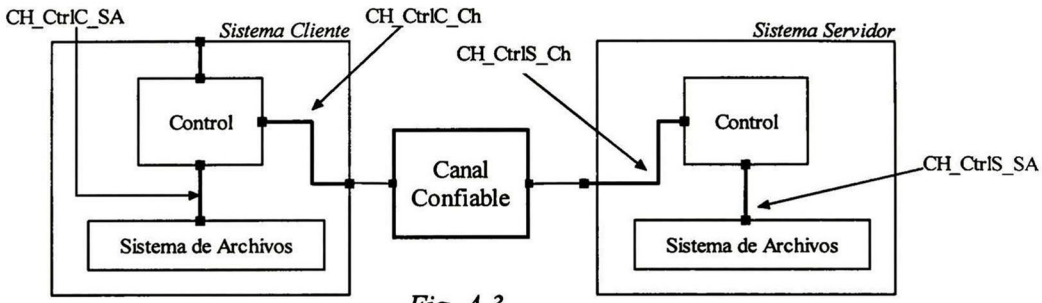


Fig. A.3

Paso 3. Mensajes.

Usuario → Sistema cliente.
- ftp(dir)
- ID
- Password
- Comando Local
- Comando Remoto
- Comando transferencia
- Quit

Sistema cliente → Usuario.
- Solicita ID
- Solicita Password
- Acceso(si)
- Acceso(no)
- Confirmación fin transferencia
- Confirmación comando local
- Confirmación comando transferencia
- Confirmación de Conexión

Control Cliente → Sistema de Archivos.
- Comando local

Sistema de archivos → Control Cliente.
- Notificación de comando local

Control cliente → Canal confiable.
- Requisición de conexión(dir)

- User(ID)
- Password(passwd)
- Comando transferencia
- Comando remoto
- Requisición de desconexión

Canal confiable → Control Cliente.
- Fin transferencia
- Acceso(si)
- Acceso(no)
- Confirmación comando remoto
- Confirmación comando transferencia
- Confirmación de conexión
- Data(buffer)

Canal Confiable → Control Servidor.
- Requisición de conexión(dir)
- User(ID)
- Password(passwd)
- Comando transferencia
- Comando remoto
- Requisición de desconexión

Control servidor → Canal Confiable.
- Fin transferencia
- Acceso(si)
- Acceso(no)
- Confirmación comando remoto
- Confirmación comando transferencia
- Confirmación de conexión
- Data(buffer)

Control servidor → Sistema de Archivos.
- Comando local

Sistema de Archivos → Control servidor
- Confirmación de comando local

Paso 4. Cronogramas.

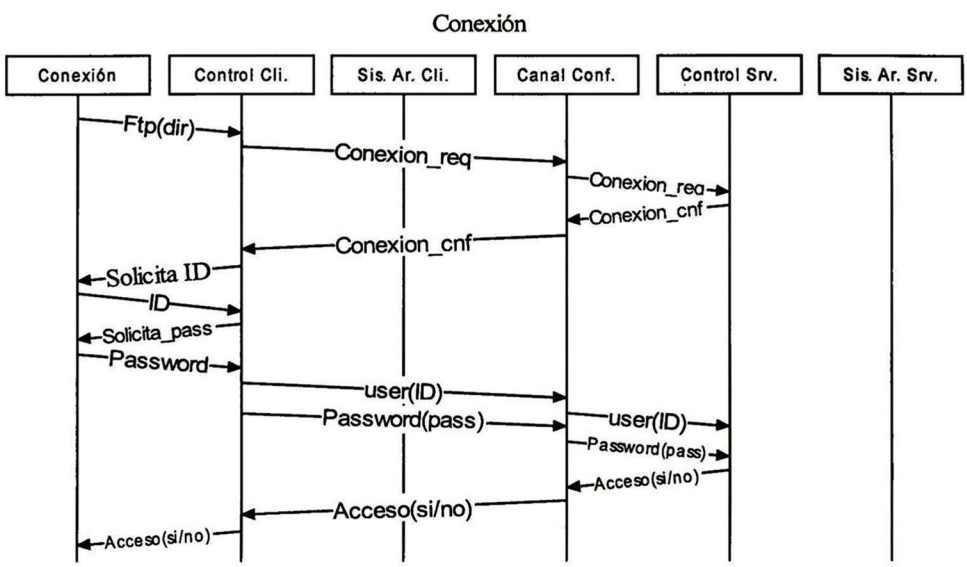


Fig. A.4

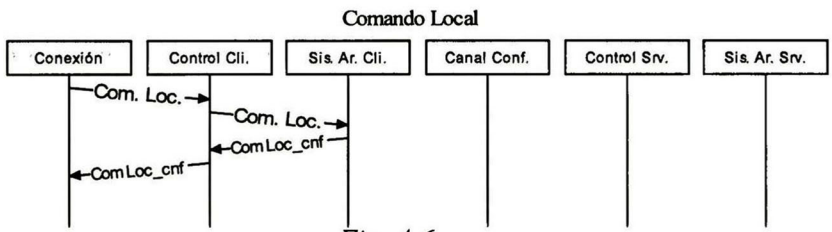


Fig. A.6

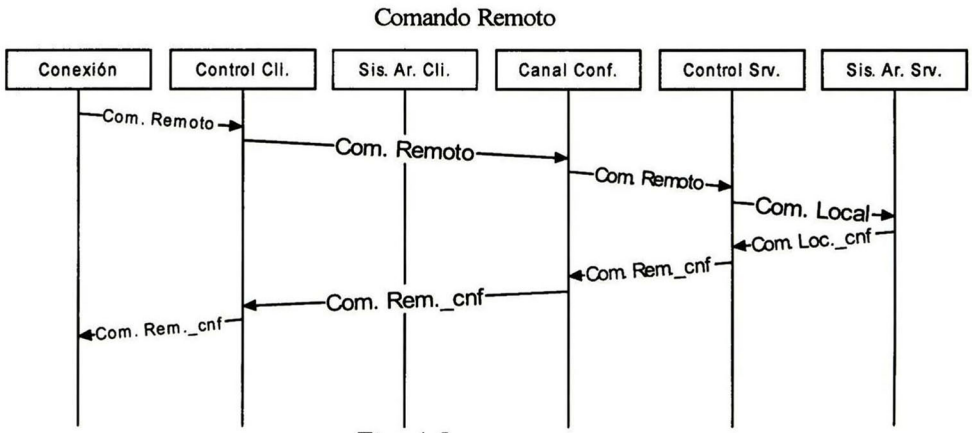


Fig. A.5

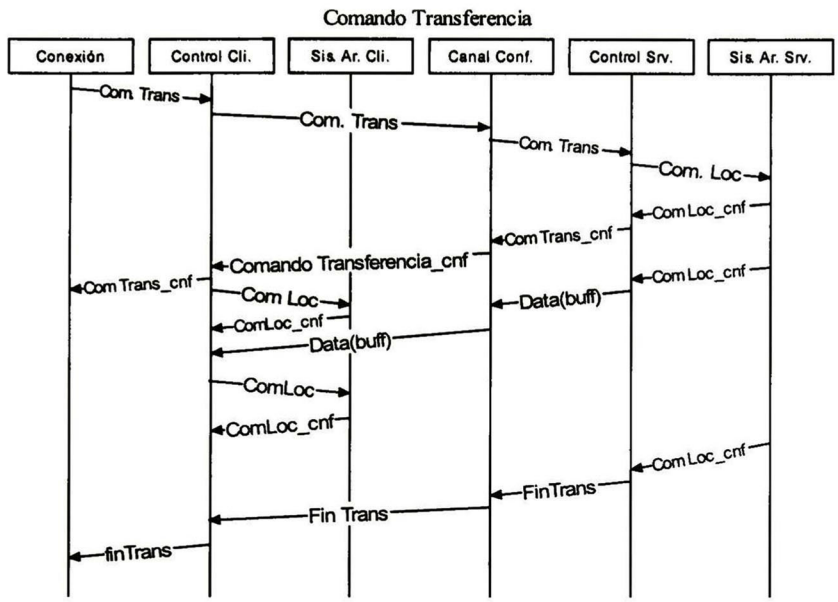

Comando Transferencia
Conexión
Control Cli.
Sis. Ar. Cli.
Canal Conf.
Control Srv.
Sis. Ar. Srv.
Com. Trans
Com. Trans
Com. Trans
Com. Loc
ComLoc_cnf
ComTrans_cnf
Comando Transferencia_cnf
ComLoc_cnf
ComTrans_cnf
Com Loc
Data(buff)
ComLoc_cnf
Data(buff)
ComLoc
ComLoc_cnf
ComLoc_cnf
FinTrans
Fin Trans
finTrans

Fig. A.7

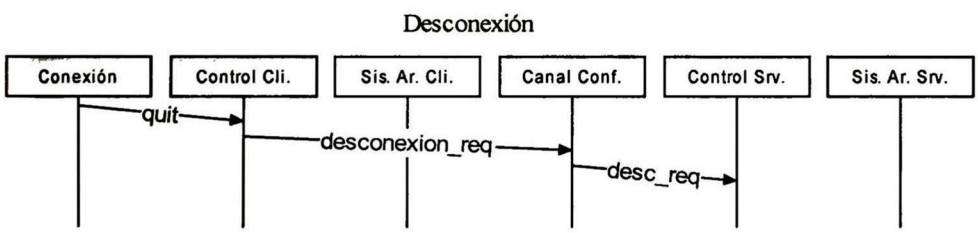
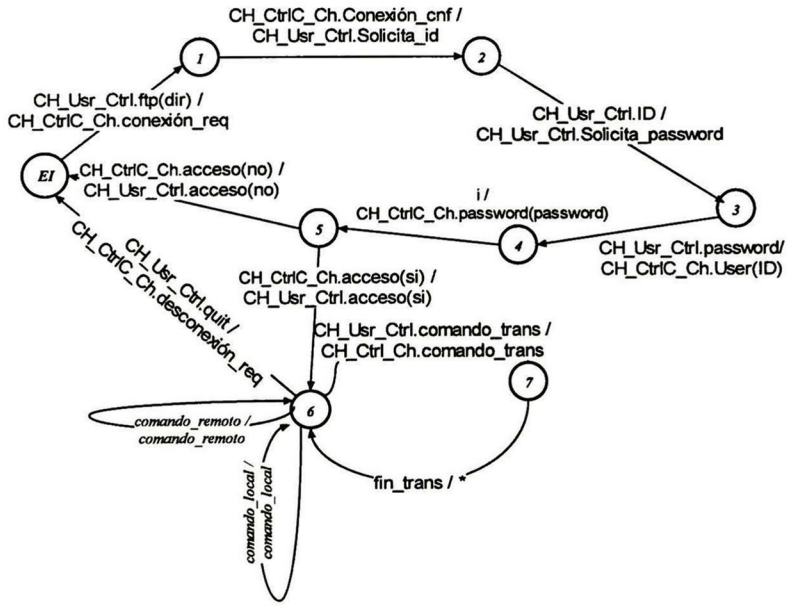

Desconexión
Conexión
Control Cli.
Sis. Ar. Cli.
Canal Conf.
Control Srv.
Sis. Ar. Srv.
quit
desconexion_req
desc_req

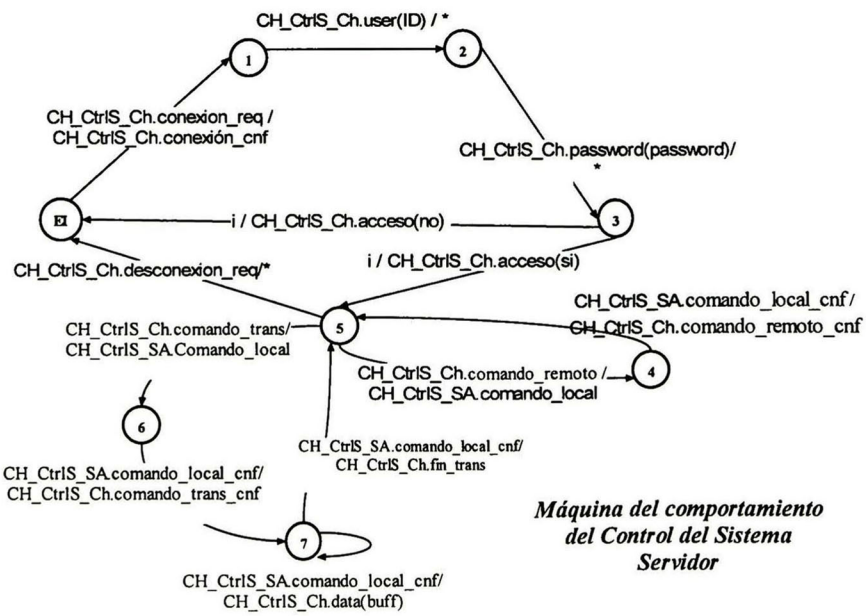
Fig. A.8

Paso 5. Autómatas.



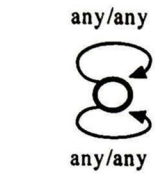
Máquina del comportamiento del Control del Sistema Cliente

Fig. A.9



Máquina del comportamiento del Control del Sistema Servidor

Fig. A.10

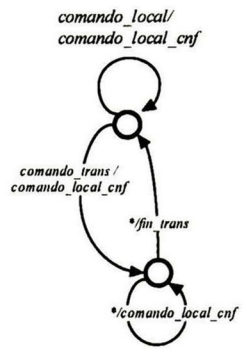


Máquina del comportamiento del canal confiable

Fig. A.11

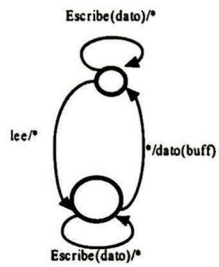
Máquina de comportamiento del Sistema de Archivos del Sistema Cliente

Fig. A.12



Máquina de comportamiento del Sistema de Archivos del Sistema Servido

Fig. A.13



Máquina del comportamiento del Buffer

Fig. A.14

Aquí nos encontramos con un problema, en el momento de la transmisión de un archivo, al control del cliente lo va almacenando en el sistema de archivos local, pero no puede enviar dos mensajes de comando local sin haber recibido la confirmación del primero. Un mensaje de data(buff) puede llegar en cualquier momento. Esto nos hace imposible modelar el comportamiento del control por medio de autómatas, bajo esta perspectiva. Proponemos entonces una solución, los mensajes que llegan de data(buff) son almacenados en un buffer. Por fines demostrativos, haremos una implementación simple de un buffer, aclarando que esta no funciona en un modelo real. En la práctica, todos los mensajes que llegan a un canal, son almacenados en un buffer y el sistema lee de este buffer.

Regresamos de esta manera a los diagramas de bloques para insertar el buffer y partimos de ahí para revisar los mensajes, los cronogramas y los autómatas. Al final, debe quedarnos un sistema que cumpla con los requerimientos dados en su totalidad. De esta manera, podemos ver como el método LAPEM es iterativo y corregible aún en la fase de autómatas.

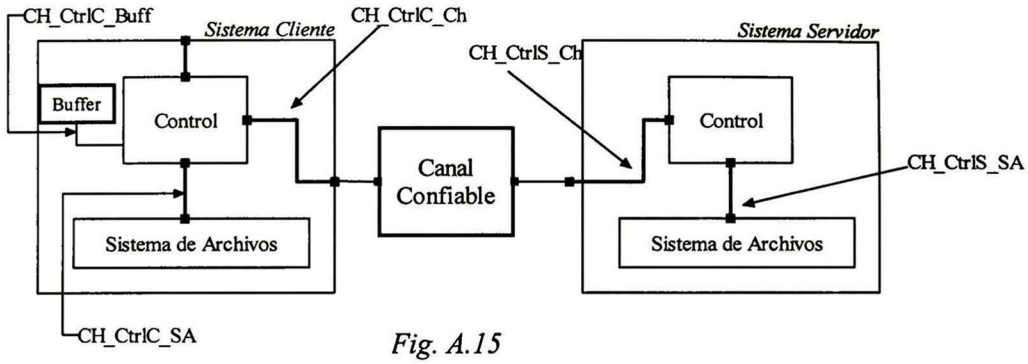


Fig. A.15

De esta manera, el sistema del cliente tiene un buffer en el que puede leer y escribir. Este buffer es una estructura de tipo FIFO.

Con esta nueva arquitectura, los mensajes quedan como en el caso anterior mas los siguientes:

Control Cliente → Buffer.
- Escribe(buffer)
- Lee

Buffer → Control Cliente.
- Dato(buffer)

El único cronograma que cambia es el de Comando Transferencia. Quedando de la siguiente manera:

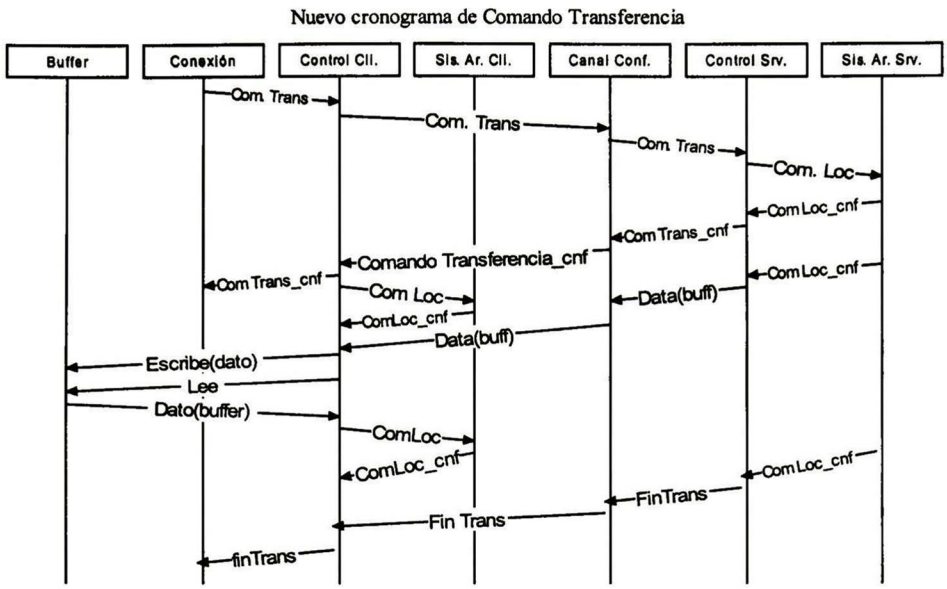
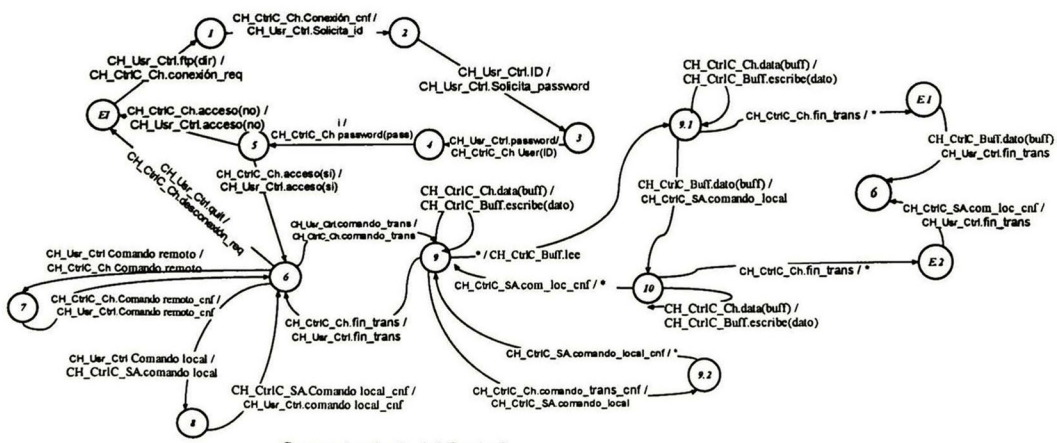


Fig. A.16

El único autómata que es modificado es el del control del cliente. Este queda de la siguiente manera:



Comportamiento del Control del Sistema Cliente

Fig. A.17

Método sugerido para la traducción de EFSM a lenguaje LOTOS

En esta sección se listan un conjunto de pasos sugeridos para efectuar la traducción de los autómatas que conforman el sistema hacia una representación usando la técnica de descripción formal LOTOS.

Paso 1. Se inicia la edición del código escribiendo lo siguiente:

```
Specification <nombre_especificación> [<compuertas del sistema>]
<Tipos de datos en ACT ONE>
Behavior
              (...(estado_inicial_M1  [...]  |||  estado_inicial_M2  [.. ])
|||



              estado_inicial_Mn[.  ])  .)
where
```

Donde los nombres de los procesos colocados inicialmente en ejecución paralelo, representan los estados iniciales de cada una de las EFSM que conforman el sistema. Los nombres tienen una nomenclatura recomendada, en la cual *estado_inicial_Mi* es el estado inicial de la máquina *i*, y donde $1 \leq i \leq n$ y *Mi* es una máquina en el modelo del sistema.

El esquema anterior representa el caso en el cual no existen comunicaciones vía compuertas entre las máquinas que conforman el sistema. Como es obvio que esta suposición no es de mucha utilidad, el esquema anterior se puede modificar a:

```
Specification <nombre_especificación> [<compuertas del sistema>]
<Tipos de datos en ACT ONE>
Behavior
  (estado_inicial_M1      [...]      |[sync_gates(M1_g1_M2,      M1_g2_M2,
   ..,M1_gn_M2)]|
  estado_inicial_M2 [. .])
  |[<sync_gates(gates(M2_g1_M3, M2_g2_M3,  ..,M2_gn_M3)>]|


  .
  |[<sync_gates(gates(Mn-1_g1_Mn, Mn-1_g2_Mn,  ..,Mn-1_gn_Mn)>]|
  estado_inicial_Mn[. .])...)
where ...
```

En este caso *<sync_gates(j, k)>* es el conjunto de compuertas que deben estar en sincronización para comunicar las máquinas M_j y M_k. Desafortunadamente la composición de esta estructura es altamente dependiente de la semántica del problema y no existen

esquemas generales para la misma. Sin embargo si existen lineamientos generales: a) hay que colocar en composición paralela los estados iniciales de cada uno de los autómatas del modelo y b) si existe en el modelo intercambio de mensajes entre dos máquinas, las compuertas involucradas en este intercambio de mensajes deben aparecer en el operador de sincronización.
Mi_gj_Mk es la *k-ésima* compuerta entre las máquinas **Mi** y **Mk**.
Recordemos además que la composición en paralelo solamente funciona por pares.

Paso 2. Expresión de la estructura de las EFSM
El siguiente paso consiste en plantear la estructura de todos los autómatas en el modelo del sistema. Para esto se crea un proceso LOTOS para cada uno de los estados de las máquinas.

Fase 2.1 Primero se procede a crear el esqueleto de todos los estados iniciales de las máquinas:

```
Process estado_inicial_M1 [<gates>] : noexit :=

Endproc

Process estado_inicial_M2 [<gates>]   noexit :=

endproc


Process estado_inicial_Mn [<gates>]   noexit :=

endproc
```

Fase 2.2 Enseguida se procede a expresar los demás estados de todas las máquinas como procesos

```
(* Expresión de los estados de la máquina 1*)
Process estado_M1_1 [<gates>](salidas:tipo, mensajes:tipo) : noexit :=

Endproc

Process estado_M1_2 [<gates>] (salidas:tipo, mensajes:tipo): noexit :=

Endproc


Process estado_M1_k1 [<gates>] (salidas:tipo, mensajes:tipo): noexit :=

Endproc

(* Expresión de los estados de la máquina 2*)
Process estado_M2_1 [<gates>] (salidas:tipo, mensajes:tipo)  noexit :=

Endproc
```

```
Process estado_M2_2 [<gates>] (salidas:tipo, mensajes:tipo)  noexit :=

Endproc


Process estado_M2_k2 [<gates>] (salidas:tipo, mensajes:tipo)  noexit :=

Endproc



(* Expresión de los estados de la máquina n*)
Process estado_Mn_1 [<gates>] (salidas:tipo, mensajes:tipo): noexit :=

Endproc

Process estado_Mn_2 [<gates>] (salidas:tipo, mensajes:tipo)  noexit :=

Endproc


Process estado_Mn_kn [<gates>] (salidas:tipo, mensajes:tipo): noexit :=

Endproc
```

Donde *estado_Mi_km* representa el m-ésimo estado de la máquina *i*. Además tenemos que $1 \leq m \leq k_i$,y *ki* es la constante representando la cantidad de estados de cada máquina la cual se caracteriza por $k_i = \circ(Q_i)$ donde Q_i es el conjunto de estados de M_i y $1 \leq i \leq n$

Paso 3. Si existe una transición del estado *j* al estado *k* de la máquina *i* esto se representa como una invocación dentro del proceso *estado_Mi_j* al proceso *estado_Mi_k.*

Fase 3.1 Si dentro del autómata *i* existen transiciones del estado *j* a los estados $k_1, k_2, ..., k_m$ donde $k_1, k_2, ..., k_m \in Q_i$. Esto se representa mediante llamadas a los procesos *process_Mi_k1*, *process_Mi_k2*,..., *process_Mi_km* relacionadas por operadores *choice* [] Si en la EFSM *i* existen condiciones lógicas de disparo en alguna transición del estado *j* a cualquier *k*, esto se representa con las guardas propias del lenguaje LOTOS.

Fase 3.2 Si en varios estados existen transiciones que lleven a un estado común, éste se declara una sola vez como proceso y se invoca desde cada estado adyacente.

Paso 4. Cuando un estado comparte variables con otro, la variable es pasada como parámetro de proceso al siguiente estado. Este paso de parámetros solo debe ocurrir entre procesos pertenecientes a la misma máquina.

Paso 5. Cuando se invoca un proceso, se le proporcionan las compuertas que vaya a utilizar así como las necesarias para los demás procesos que mandará a llamar. Por esto es recomendable pasar todas las compuertas especificadas en el modelo estructural para la máquina *i* a cada uno de los procesos que representan estados de la máquina *i*.

Paso 6. Dado que el modelo en LOTOS representa EFSMs, en muchas ocasiones serán necesarias variables internas para cada máquina. Para estos casos el lenguaje LOTOS cuenta dentro de su estructura con el leguaje de especificación de tipos de datos abstractos (TDAs) ACT ONE

Paso 7. La comunicación entre EFSMs se hará solamente mediante el envío de mensajes a través de las compuertas que comunican a los módulos.

Código.

```
SPECIFICATION Ejemplo [CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA, CH_CtrlS_SA,
CH_CtrlS_Ch]: NOEXIT


TYPE Boolean IS
   SORTS
         bool
   OPNS
         true   false                                          -> bool
         not                                        bool       -> bool
         _and_   _or_ , _xor_ , _implies_
         _iff_   _eqbool_   _ne_                    bool, bool -> bool
   EQNS
         FORALL x   y   bool
         OFSORT bool
                not(not(x))      x;
                not(true)        false;
                not(false)       true;
                x and x          x;
                x and true       x;
                true and x       x;
                x or y           not(not(x) and not(y));
                x xor y          (x and not(y)) or (y and not(x));
                x implies y      y or not(x);
                x iff y          (y or not(x)) and (x or not(y));
(*          x ne y          not(x iff y);*)
                x eqbool y       x iff y;
ENDTYPE


TYPE natural IS boolean
   SORTS nat
   OPNS
        0                        -> nat
        1                        -> nat
        2                        -> nat
        3                        -> nat
        4                        -> nat
        5                        -> nat
        6                        -> nat
        7                        -> nat
        8                        -> nat
        9                        -> nat
        inc                      nat -> nat
```

```
        _mod_ , _-_ , _*_, _+_    nat,nat -> nat
        _eq_ ,_ne_, _lt_   _gt_   _le_   _ge_   nat   nat -> bool
   EQNS
        FORALL x,y:nat
          OFSORT bool
                0 eq 0   true;
                1 eq 1   true;
                2 eq 2   true;
                3 eq 3   true;
                4 eq 4 = true;
                5 eq 5 = true;
                6 eq 6 = true;
                7 eq 7 = true;
                8 eq 8 = true;
                9 eq 9 = true;
                0 eq inc(x)   false;
                inc(x) eq 0   false;
                inc(x) eq inc(y)   x eq y;
                x ne y = not (x eq y);
                0 lt 0 = false;
                inc(x) lt inc(y)   x lt y;
                0 lt inc(x)   true;
                inc(x) lt 0   false;
                x gt y   not((x eq y) or (x lt y));
                x ge y   not(x lt y);
                x le y   not(x gt y);

          OFSORT nat
          0+x = x;
          inc(x)+y   x+inc(y);
          (*inc(x)+y   inc(x+y);*)
          0*x = 0;
          inc(x)*y   (x*y)+y;
          0-x = 0;
          x-x = 0;
          inc(x)-inc(y)   x-y;
          x-0   x;
                x lt y => x mod y   x;
                not(x lt y) => x mod y = (x-y) mod y
ENDTYPE

TYPE Id IS Boolean
  SORTS
        Id        (* identifiers process *)
  OPNS
        A,B,C,D,E          :          -> Id
        _equal_        :Id,Id   ->  bool
   EQNS
        OFSORT Bool
           (A equal A)   true  ; (B equal A)   false ; (C equal A)   false ;
           (D equal A)   false ; (E equal A)   false ; (A equal B)   false ;
           (B equal B)   true    (C equal B)   false ; (D equal B)   false ;
           (E equal B)   false ; (A equal C)   false ; (B equal C)   false ;
           (C equal C)   true  ; (D equal C)   false ; (E equal C)   false ;
           (A equal D)   false ; (B equal D)   false ; (C equal D)   false ;
           (D equal D)   true  ; (E equal D)   false ; (A equal E)   false ;
           (B equal E)   false ; (C equal E) = false ; (D equal E) = false ;
           (E equal E)   true  ;

ENDTYPE


BEHAVIOR

((((Estado_Inicial_Control_Cte[CH_Usr_Ctrl,   CH_CtrlC_Ch,   CH_CtrlC_Buff,   CH_CtrlC_SA](0)
|[CH_Usr_Ctrl]|

        Estado_Inicial_Usuario[CH_Usr_Ctrl])   |[CH_CtrlC_Buff]|
        Estado_Inicial_Buffer[CH_CtrlC_Buff](0)) |[CH_CtrlC_SA]|
        Estado_Inicial_Sistema_ArchC[CH_CtrlC_SA](0)) |[CH_CtrlC_Ch]|
        Estado_Inicial_Canal_Conf[CH_CtrlC_Ch, CH_CtrlS_Ch])
```

```
        |[CH_CtrlS_Ch]|

        (Estado_Inicial_Control_Serv[CH_CtrlS_Ch, CH_CtrlS_SA](0)   |[CH_CtrlS_SA]|
        Estado_Inicial_Sistema_ArchS[CH_CtrlS_SA](0))


WHERE

        PROCESS     Estado_Inicial_Control_Cte[CH_Usr_Ctrl,     CH_CtrlC_Ch,     CH_CtrlC_Buff,
CH_CtrlC_SA](connection_req:nat):NOEXIT:=
                CH_Usr_Ctrl ? ftp_dir :nat;
                CH_CtrlC_Ch ! connection_req;
                Control_Cte_1[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1)
        ENDPROC

        PROCESS         Control_Cte_1[CH_Usr_Ctrl,         CH_CtrlC_Ch,         CH_CtrlC_Buff,
CH_CtrlC_SA](solicita_id:nat):NOEXIT:=
                CH_CtrlC_Ch ? conexion_cnf   nat;
                CH_Usr_Ctrl ! solicita_id;
                Control_Cte_2[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](2)

        ENDPROC

        PROCESS         Control_Cte_2[CH_Usr_Ctrl,         CH_CtrlC_Ch,         CH_CtrlC_Buff,
CH_CtrlC_SA](solicita_passwd:nat):NOEXIT:=
                CH_Usr_Ctrl ? ID :nat;
                CH_Usr_Ctrl !solicita_passwd;
                Control_Cte_3[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](ID)

        ENDPROC

        PROCESS         Control_Cte_3[CH_Usr_Ctrl,         CH_CtrlC_Ch,         CH_CtrlC_Buff,
CH_CtrlC_SA](ID:nat):NOEXIT:=
                CH_Usr_Ctrl ? password :nat;
                CH_CtrlC_Ch ! ID;
                Control_Cte_4[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](password)

        ENDPROC

        PROCESS  Control_Cte_4[CH_Usr_Ctrl,  CH_CtrlC_Ch,  CH_CtrlC_Buff,  CH_CtrlC_SA](pass
:nat):NOEXIT:=
                CH_CtrlC_Ch ! pass;
                Control_Cte_5[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA]

        ENDPROC

        PROCESS Control_Cte_5[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA]:NOEXIT:=
                CH_CtrlC_Ch ? acceso :nat;(
                ([acceso  eq  0]  ->  Estado_Inicial_Control_Cte[CH_Usr_Ctrl,  CH_CtrlC_Ch,
CH_CtrlC_Buff, CH_CtrlC_SA](0))
                []
                ([acceso  eq  1]  ->  Control_Cte_6[CH_Usr_Ctrl,  CH_CtrlC_Ch,  CH_CtrlC_Buff,
CH_CtrlC_SA](0,2,1))
                )

        ENDPROC

        PROCESS         Control_Cte_6[CH_Usr_Ctrl,         CH_CtrlC_Ch,         CH_CtrlC_Buff,
CH_CtrlC_SA](comando_local:nat, comando_remoto:nat, comando_trans:nat):NOEXIT:=

                CH_Usr_Ctrl ? msg :nat;(
                ([(msg eq 2)] ->CH_CtrlC_SA !comando_local;
                Control_Cte_8[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA])
                []
                ([(msg eq 3)] ->CH_CtrlC_Ch !comando_remoto;
                Control_Cte_7[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA])
                []
                ([(msg eq 4)] ->CH_CtrlC_Ch !comando_trans;
                Control_Cte_9[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1,2,0))
                []
```

```
            ([(msg eq 5)]->CH_CtrlC_Ch !3;(********   3= comando quit   ************)
            Estado_Inicial_Control_Cte[CH_Usr_Ctrl,      CH_CtrlC_Ch,      CH_CtrlC_Buff,
CH_CtrlC_SA](0))
            )
      ENDPROC

      PROCESS Control_Cte_7[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA]:NOEXIT:=
            CH_CtrlC_Ch ? comando_remoto_cnf :nat;
            CH_Usr_Ctrl ! comando_remoto_cnf;
            Control_Cte_6[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](0,2,1)

      ENDPROC

      PROCESS Control_Cte_8[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA]:NOEXIT:=
            CH_CtrlC_SA ? comando_local_cnf :nat;
            CH_Usr_Ctrl ! comando_local_cnf;
            Control_Cte_6[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](0,2,1)

      ENDPROC

      PROCESS        Control_Cte_9[CH_Usr_Ctrl,        CH_CtrlC_Ch,        CH_CtrlC_Buff,
CH_CtrlC_SA](escribe:nat, lee:nat, comando_local:nat):NOEXIT:=

            CH_CtrlC_Ch ?msg:nat;
            (
            (
            [msg eq 0]->((******  llega un buffer de datos    ***********)
                  CH_CtrlC_Buff !escribe;
                  Control_Cte_9[CH_Usr_Ctrl,        CH_CtrlC_Ch,        CH_CtrlC_Buff,
CH_CtrlC_SA](1,2,0)
                  )
            )
            []
            (
            [msg eq 1]->((******** llega comando_trans_cnf    ***********)
                  CH_CtrlC_SA ! comando_local;
                  Control_Cte_9_2[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA]
                  )
            )
            []
            (
            [msg eq 2]->((******** llega fin_trans    ***********)
                  CH_Usr_Ctrl ! 0;
                  Control_Cte_6[CH_Usr_Ctrl,        CH_CtrlC_Ch,        CH_CtrlC_Buff,
CH_CtrlC_SA](0,2,1))
            )
            )
            []
            (
            CH_CtrlC_Buff ! lee;
            Control_Cte_9_1[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1,0))


      ENDPROC

      PROCESS        Control_Cte_9_1[CH_Usr_Ctrl,        CH_CtrlC_Ch,        CH_CtrlC_Buff,
CH_CtrlC_SA](escribe:nat, comando_local:nat):NOEXIT:=

            (CH_CtrlC_Buff ?dato:nat;
            CH_CtrlC_SA ! Comando_local;
            Control_Cte_10[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1))
            []
            (CH_CtrlC_Ch ? dato :nat;
                  (
                  ([dato eq 0] ->CH_CtrlC_Buff !escribe;(****  llego dato  **********)
                  Control_Cte_9_1[CH_Usr_Ctrl,        CH_CtrlC_Ch,        CH_CtrlC_Buff,
CH_CtrlC_SA](1,0))
                  []
                  [dato eq 2] ->Control_Cte_E1[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA]
                  (******   llega fin trans    ************)
```

```
                )
        )
ENDPROC

PROCESS Control_Cte_9_2[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA]:NOEXIT:=
        CH_CtrlC_SA ? comando_local_cnf :nat;
        Control_Cte_9[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1,2,0)

ENDPROC

PROCESS Control_Cte_10[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](escribe
:nat):NOEXIT:=
        (CH_CtrlC_Ch ? dat:nat;
                (
                ([dat eq 0]-> CH_CtrlC_Buff !escribe;
                Control_Cte_10[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA](1))
                []
                [dat eq 2]->Control_Cte_E2[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA]
                )
        )
        []
        (CH_CtrlC_SA ? Comando_local_cnf :nat;
        Control_Cte_9[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](1,2,0))

ENDPROC

PROCESS Control_Cte_E1[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA]:NOEXIT:=
        CH_CtrlC_Buff ?dato:nat;
        CH_Usr_Ctrl ! 0;
        Control_Cte_6[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](0,2,1)

ENDPROC

PROCESS Control_Cte_E2[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff,
CH_CtrlC_SA]:NOEXIT:=
        CH_CtrlC_SA ? Comando_local_cnf :nat;
        CH_Usr_Ctrl ! 0;
        Control_Cte_6[CH_Usr_Ctrl, CH_CtrlC_Ch, CH_CtrlC_Buff, CH_CtrlC_SA](0,2,1)

ENDPROC

(*************     control del servidor       ************************)

PROCESS Estado_Inicial_Control_Serv[CH_CtrlS_Ch, CH_CtrlS_SA](conexion_cnf
:nat):NOEXIT:=
        CH_CtrlS_Ch ? conexion_req :nat;
        CH_CtrlS_Ch ! conexion_cnf;
        Control_Serv_1[CH_CtrlS_Ch, CH_CtrlS_SA]

ENDPROC

PROCESS Control_Serv_1[CH_CtrlS_Ch, CH_CtrlS_SA]:NOEXIT:=
        CH_CtrlS_Ch ? ID:nat;
        Control_Serv_2[CH_CtrlS_Ch, CH_CtrlS_SA](ID)

ENDPROC

PROCESS Control_Serv_2[CH_CtrlS_Ch, CH_CtrlS_SA](ID:nat):NOEXIT:=
        CH_CtrlS_Ch ? Pass:nat;
        Control_Serv_3[CH_CtrlS_Ch, CH_CtrlS_SA](ID, Pass)

ENDPROC
```

```
        PROCESS Control_Serv_3[CH_CtrlS_Ch, CH_CtrlS_SA](ID:nat, Pass:nat):NOEXIT:=
                i;(
                (CH_CtrlS_Ch ! 0;(****    niega la entrada       ********)
                Estado_Inicial_Control_Serv[CH_CtrlS_Ch, CH_CtrlS_SA](0))
                []
                (CH_CtrlS_Ch ! 1;(****    acepta la entrada      ********)
                Control_Serv_5[CH_CtrlS_Ch, CH_CtrlS_SA](0))
                )

        ENDPROC

        PROCESS Control_Serv_4[CH_CtrlS_Ch, CH_CtrlS_SA](comando_remoto_cnf :nat):NOEXIT:=
                CH_CtrlS_SA ?comando_local_cnf :nat;
                CH_CtrlS_Ch ! comando_remoto_cnf;
                Control_Serv_5[CH_CtrlS_Ch, CH_CtrlS_SA](0)


        ENDPROC

        PROCESS Control_Serv_5[CH_CtrlS_Ch, CH_CtrlS_SA](comando_local :nat):NOEXIT:=
                CH_CtrlS_Ch ?msg :nat;(
                ([msg eq 1] ->CH_CtrlS_SA !1;(******llega comando transferencia********)
                Control_Serv_6[CH_CtrlS_Ch, CH_CtrlS_SA](1))
                []
                ([msg   eq   2]   ->CH_CtrlS_SA   !comando_local;(*******llega   un   comando
remoto****************)
                Control_Serv_4[CH_CtrlS_Ch, CH_CtrlS_SA](0))
                []
                ([(msg      eq      3)]      ->      Estado_Inicial_Control_Serv[CH_CtrlS_Ch,
CH_CtrlS_SA](0))(****comando quit***)
                )

        ENDPROC

        PROCESS Control_Serv_6[CH_CtrlS_Ch, CH_CtrlS_SA](comando_trans_cnf :nat):NOEXIT:=

                CH_CtrlS_SA ?comando_local_cnf :nat;
                CH_CtrlS_Ch !comando_trans_cnf;
                Control_Serv_7[CH_CtrlS_Ch, CH_CtrlS_SA](0,2)

        ENDPROC

        PROCESS Control_Serv_7[CH_CtrlS_Ch, CH_CtrlS_SA](buff :nat, fin_trans:nat):NOEXIT:=

                CH_CtrlS_SA ?comando_local_cnf :nat;(

                ([comando_local_cnf eq 0]->(******   llega un dato    ***********)
                CH_CtrlS_Ch ! buff;
                Control_Serv_7[CH_CtrlS_Ch, CH_CtrlS_SA](0,2))
                []
                ([comando_local_cnf eq 1]->(*******   llega un fin archivo    ***********)
                CH_CtrlS_Ch ! fin_trans;
                Control_Serv_5[CH_CtrlS_Ch, CH_CtrlS_SA](0))
                )

        ENDPROC

(*****************      Autómata de canal CONFIABLE      ***************************)



        PROCESS Estado_Inicial_Canal_Conf[CH_CtrlC_Ch, CH_CtrlS_Ch]:NOEXIT:=
                CH_CtrlS_Ch ?msgS :nat;
                CH_CtrlC_Ch ! msgS;
                Estado_Inicial_Canal_Conf[CH_CtrlC_Ch, CH_CtrlS_Ch]
                []
                CH_CtrlC_Ch ? msgC :nat;
                CH_CtrlS_Ch ! msgC;
                Estado_Inicial_Canal_Conf[CH_CtrlC_Ch, CH_CtrlS_Ch]
```

```
        ENDPROC

(*****************      Autómata de Sistema de Archivos Cliente      *****************)

        PROCESS Estado_Inicial_Sistema_ArchC[CH_CtrlC_SA](comando_local_cnf:nat):NOEXIT:=
        CH_CtrlC_SA ?comando_local:nat;
        CH_CtrlC_SA !comando_local_cnf;
        Estado_Inicial_Sistema_ArchC[CH_CtrlC_SA](0)

        ENDPROC


(*****************      Autómata de Sistema de Archivos Servidor      *****************)

        PROCESS Estado_Inicial_Sistema_ArchS[CH_CtrlS_SA](comando_local_cnf:nat):NOEXIT:=
        CH_CtrlS_SA ?comando_local:nat;
        CH_CtrlS_SA !comando_local_cnf;(
        [comando_local eq 0]-> Estado_Inicial_Sistema_ArchS[CH_CtrlS_SA](0)
        []
        [comando_local eq 1] ->Sistema_ArchS_1[CH_CtrlS_SA]
        )

        ENDPROC


        PROCESS Sistema_ArchS_1[CH_CtrlS_SA]:NOEXIT:=
        (CH_CtrlS_SA ! 0;(*****    regresa un dato al servidor    *********)
        Sistema_ArchS_1[CH_CtrlS_SA])
        []
(       CH_CtrlS_SA !1;(********   regresa fin de archivo al serv   *********)
        Estado_Inicial_Sistema_ArchS[CH_CtrlS_SA](0))

        ENDPROC


(*****************      Autómata de BUFFER      ****************************)

        PROCESS Estado_Inicial_Buffer[CH_CtrlC_Buff](buff:nat):NOEXIT:=
                CH_CtrlC_Buff ?msg:nat;
                ([(msg eq 1)] ->Estado_Inicial_Buffer[CH_CtrlC_Buff](0)(** escribe  *****)
                []
                [(msg eq 2)] -> Buffer_1[CH_CtrlC_Buff])(*** lee un dato ****)


        ENDPROC


        PROCESS Buffer_1[CH_CtrlC_Buff]:NOEXIT:=
                (CH_CtrlC_Buff ?msg:nat;
                Buffer_1[CH_CtrlC_Buff](***  llega mensaje escribe  **)
                )
                []
                (CH_CtrlC_Buff !0;(**  sale mensaje dato  **)
                Estado_Inicial_Buffer[CH_CtrlC_Buff](0)
                )

        ENDPROC

(*****************      Autómata de Usuario      ****************************)

        PROCESS Estado_Inicial_Usuario[CH_Usr_Ctrl]:NOEXIT:=
```

```
        (
                CH_Usr_Ctrl ! 0;(*******    Envia comando ftp(dir)      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 2;(*******    Envia comando local      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 3;(*******    Envia comando remoto      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 4;(*******    Envia comando trans      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 5;(*******    Envia comando quit      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 8;(*******    Envia ID      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
                []
                CH_Usr_Ctrl ! 9;(*******    Envia Password      *******)
                Estado_Inicial_Usuario[CH_Usr_Ctrl]
        )
        []
        (
        CH_Usr_Ctrl ? msg_in :nat;
        Estado_Inicial_Usuario[CH_Usr_Ctrl]
        )

    ENDPROC

ENDSPEC
```

Simulación.

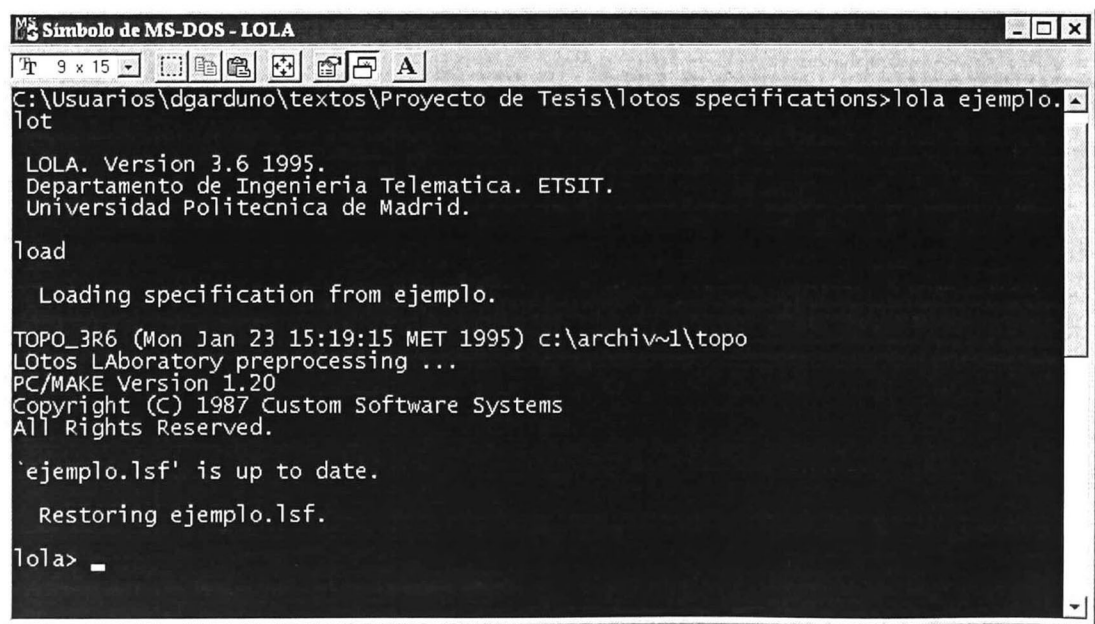


Fig. A.18

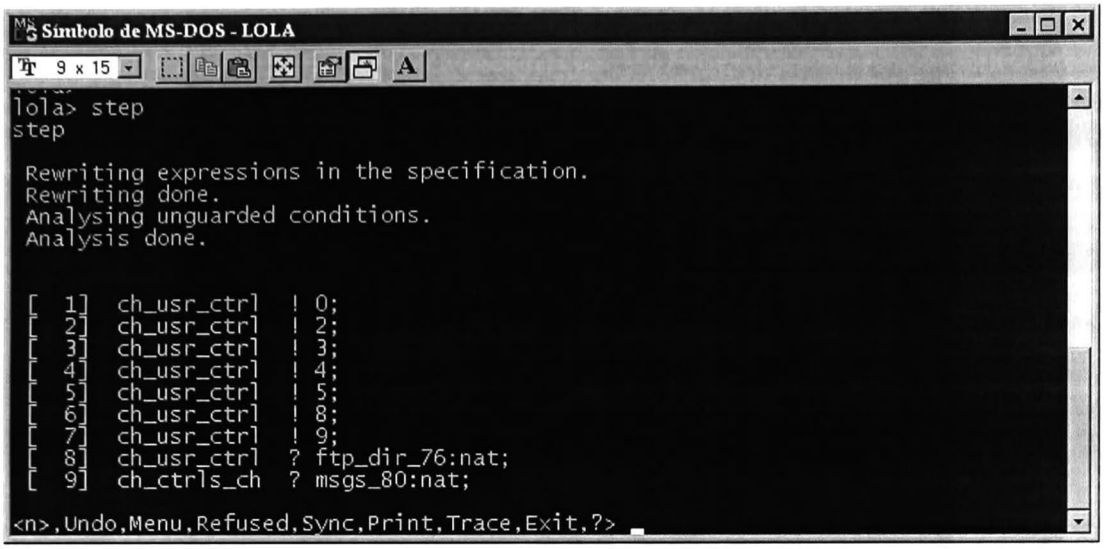


Fig. A.19

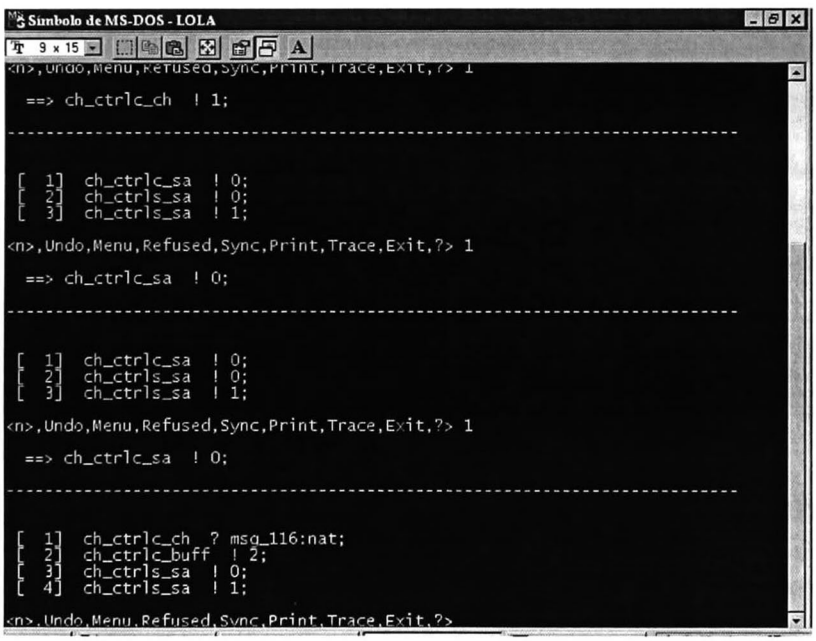


Fig. A.20

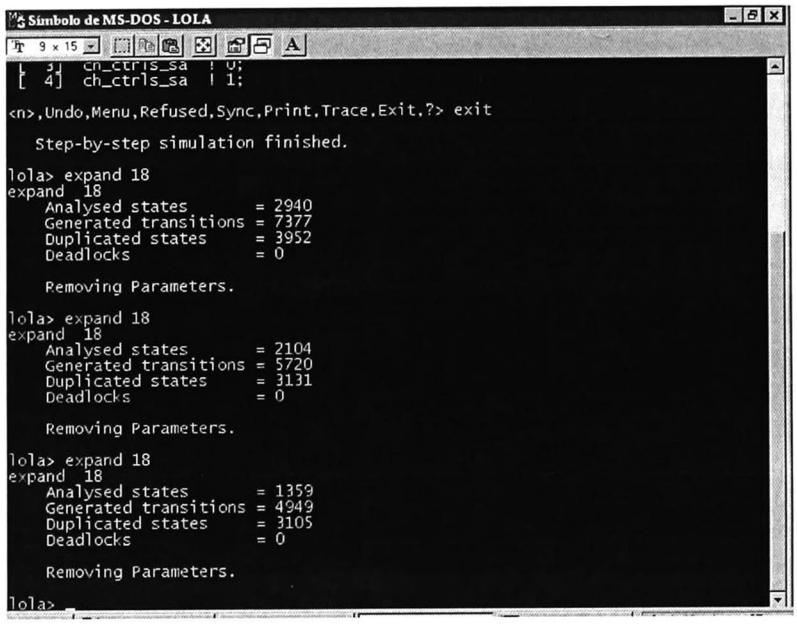


Fig. A.21

Implementación.

La codificación en lenguaje C, partiendo de una especificación en LOTOS, puede ser hecha mediante alguna herramienta como TOPO. La codificación en C hecha para este ejemplo, será hecha utilizando esta herramienta.

Reestructuración.

Para el mejor entendimiento del sistema, dividiremos los autómatas por acciones: conexión aceptada, conexión rechazada, ejecución de comando local, ejecución de comando remoto, ejecución de comando transferencia y desconexión. Mostraremos también un grafo de observabilidad (del usuario).

Grafos para conexión rechazada.

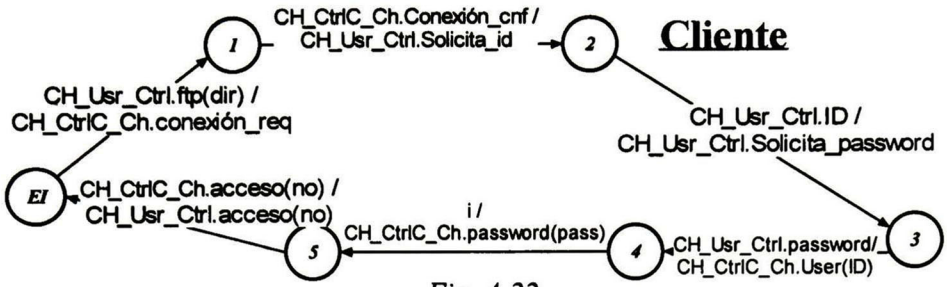


Fig. A.22

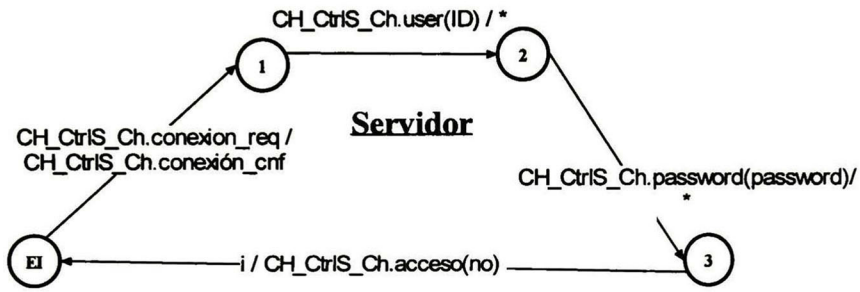


Fig. A.23

Grafos de conexión aceptada.

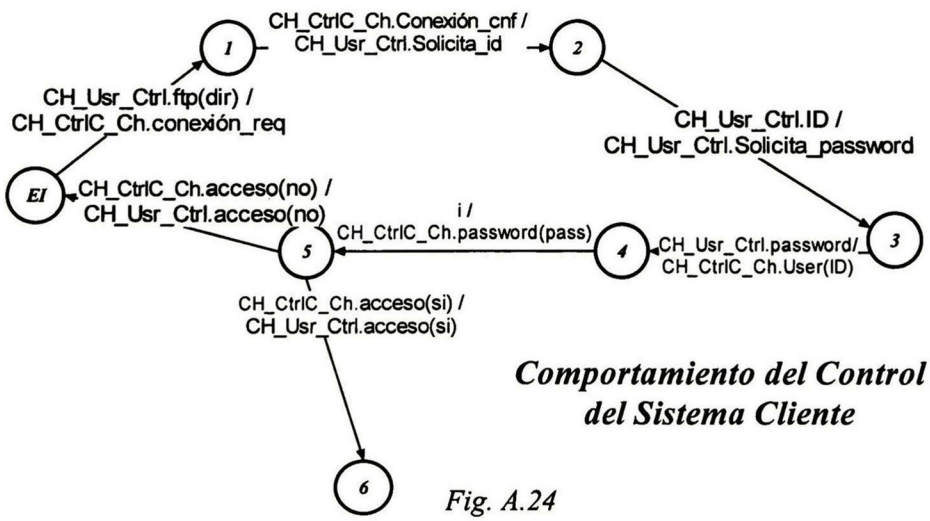
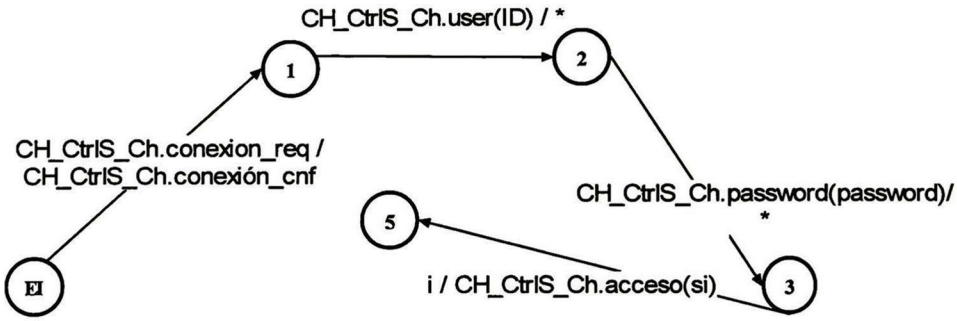


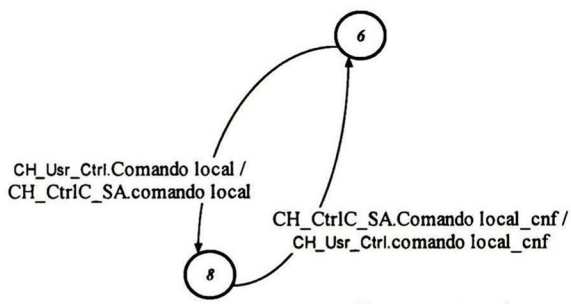
Fig. A.24



Máquina del comportamiento del Control del Sistema Servidor

Fig. A.25

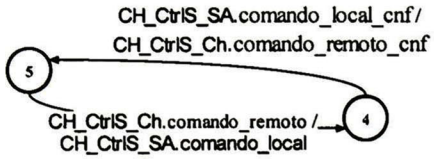
Grafo de conexión ejecución de comando local.



Comportamiento del Control del Sistema Cliente

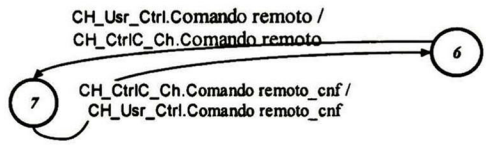
Fig. A.26

Ejecución de comando Remoto.



Máquina del comportamiento del Control del Sistema Servidor

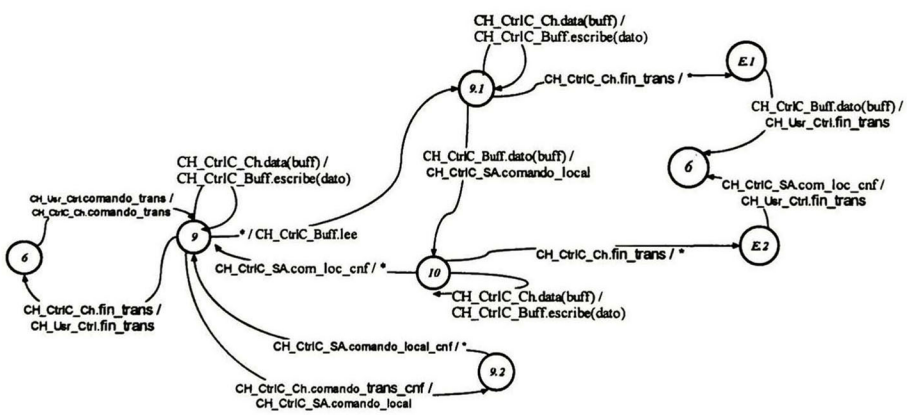
Fig. A.27



Comportamiento del Control del Sistema Cliente

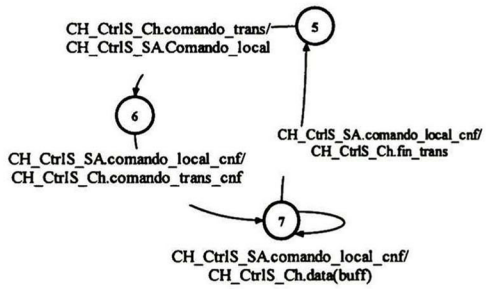
Fig. A.28

Grafos ejecución de comando transferencia.



Comportamiento del Control del Sistema Cliente

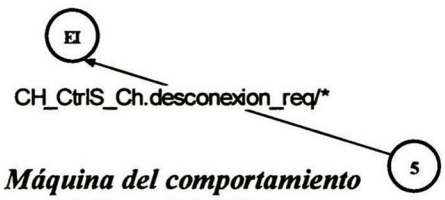
Fig. A.29



Máquina del comportamiento del Control del Sistema Servidor

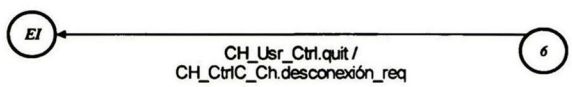
Fig. A.30

Grafos de Descomexión.



Máquina del comportamiento del Control del Sistema Servidor

Fig. A.31



Comportamiento del Control del Sistema Cliente

Fig. A.32

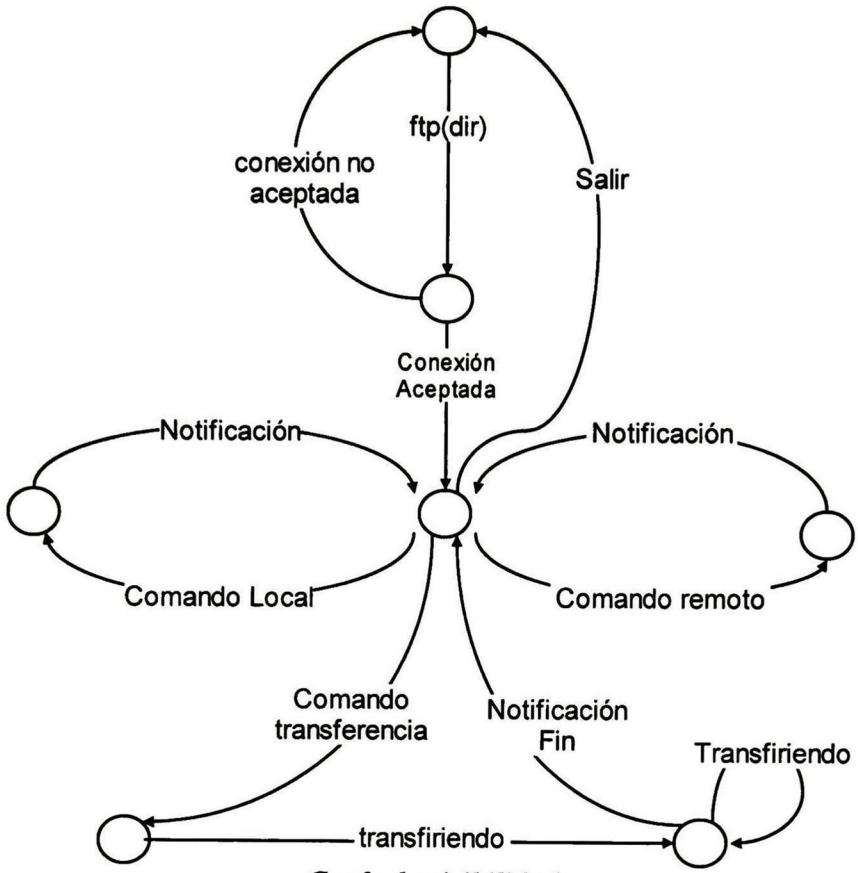


Fig. A.33

Grafo de Visibilidad del usuario.

Apéndice B.

Código LOTOS de la Arquitectura Propuesta

```
(*************************************************************************)
(***                                                                   ****)
(***   Simulación del sistema de control, trasmisión y algoritmos de   ****)
(***   sincronización para el sistema VOIP                             ****)
(***                                                                   ****)
(***                                                                   ****)
(***                                                                   ****)
(***    David Garduño, Guadalajara, Jal. México.                       ****)
(***                   Diciembre 1999                                  ****)
(***                                                                   ****)
(*************************************************************************)
(*************************************************************************)
(*************************************************************************)


SPECIFICATION transmisor_VoIP[CH_DB_C_1, CH_U_C_1, CH_C_MM_1, CH_MM_H_1, CH_MM_M_1,
                              CH_C_H_1, CH_H_C0_1, CH_H_Co_1, CH_C0_N_1, CH_Co_N_1,
                              CH_DB_C_2, CH_U_C_2, CH_C_MM_2, CH_MM_H_2, CH_MM_M_2,
                              CH_C_H_2, CH_H_C0_2, CH_H_Co_2, CH_C0_N_2, CH_Co_N_2]:NOEXIT

(*************************************************************************)
(**    TIPOS DE DATOS                                                   **)
(*************************************************************************)


(*************     Tipo de datos   BOOLEAN            *************)

Type Boolean is
  SORTS
    Bool
  OPNS
    true (*| constructor |*), false (*| constructor |*): -> Bool
    not (*| nonconstructor |*): Bool -> Bool
    _ and _ (*| nonconstructor |*),
    _ or _ (*| nonconstructor |*),
    _ xor _ (*| nonconstructor |*),
    _ implies _ (*| nonconstructor |*),
    _ iff _ (*| nonconstructor |*)       Bool, Bool -> Bool
    _ eq _ (*| nonconstructor |*),
    _ ne _ (*| nonconstructor |*)   Bool, Bool -> Bool
  EQNS
    FORALL x, y: Bool
    OFSORT Bool
      not (true OF Bool) = false OF Bool ;
      not (false OF Bool)   true OF Bool ;
      x and true OF Bool = x ;
      true OF Bool and x = x ;
      x and false OF Bool   false OF Bool ;
      false OF Bool and x   false OF Bool ;
      x or true OF Bool   true OF Bool ;
      true OF Bool or x   true OF Bool ;
      x or false OF Bool = x ;
      false OF Bool or x = x ;
      x xor y   (x and not (y)) or (y and not (x)) ;
      x implies y = y or not (x) ;
      x iff y   (x implies y) and (y implies x) ;
      x eq y   x iff y ;
      x ne y   x xor y ;
  ENDTYPE
```

```
(*************      Tipo de datos   NATURAL            *************)

TYPE natural IS boolean
   SORTS nat
   OPNS
        0                          -> nat
        1                          -> nat
        2                          -> nat
        3                          -> nat
        4                          -> nat
        5                          -> nat
        6                          -> nat
        7                          -> nat
        8                          -> nat
        9                          -> nat
        inc                        nat -> nat
        _mod_   _-_   _*_, _+_     nat,nat -> nat
        _eq_ ,_ne_, _lt_   _gt_   _le_   _ge_   nat   nat -> bool

   EQNS
                FORALL x,y:nat
                OFSORT bool
                        0 eq 0   true;
                        1 eq 1   true;
                        2 eq 2   true;
                        3 eq 3   true;
                        4 eq 4   true;
                        5 eq 5   true;
                        6 eq 6   true;
                        7 eq 7   true;
                        8 eq 8   true;
                        9 eq 9   true;
                        0 eq inc(x)   false;
                        inc(x) eq 0   false;
                        inc(x) eq inc(y) = x eq y;
                        x ne y   not (x eq y);
                        0 lt 0 = false;
                        inc(x) lt inc(y)   x lt y;
                        0 lt inc(x)   true;
                        inc(x) lt 0 = false;
                        x gt y = not((x eq y) or (x lt y));
                        x ge y = not(x lt y);
                        x le y = not(x gt y);

        OFSORT nat
                0+x   x;
                inc(x)+y = x+inc(y);
                (*inc(x)+y = inc(x+y);*)
                0*x   0;
                inc(x)*y = (x*y)+y;
                0-x   0;
                x-x   0;
                inc(x)-inc(y) = x-y;
                x-0   x;
                x lt y => x mod y   x;
                not(x lt y) => x mod y   (x-y) mod y
ENDTYPE



(*************      Tipo de datos   MSG            *************)

TYPE msg IS Boolean
        SORTS   msg

        OPNS
                open_connection          -> msg
```

```
        close_connection, connection_ready          -> msg
        connection_accepted, connection_rejected            -> msg
        connection_opened, connection_closed   -> msg
        init, start, stopp      -> msg
        alias_to_IP, IP_to_alias, response     -> msg
        connection_req, C_C_exchange, media_unit     : -> msg
        _eq_, _neq_    :msg,msg       ->bool

    EQNS
    OFSORT Bool
        (open_connection eq open_connection) = true;
        (close_connection eq close_connection) = true;
        (connection_ready eq connection_ready) = true;
        (connection_accepted eq connection_accepted) = true;
        (connection_rejected eq connection_rejected)   true;
        (connection_opened eq connection_opened) = true;
        (connection_closed eq connection_closed) = true;
        (init eq init) = true;
        (start eq start)   true;
        (stopp eq stopp)   true;
        ( alias_to_IP eq alias_to_IP) = true;
        (IP_to_alias eq IP_to_alias)   true;
        (response eq response)   true;
        (connection_req eq connection_req) = true;
        (C_C_exchange eq C_C_exchange)   true;
        (media_unit eq media_unit) = true;


        (open_connection neq open_connection) = false;
        (close_connection neq close_connection)   false;
        (connection_ready neq connection_ready)   false;
        (connection_accepted neq connection_accepted)   false;
        (connection_rejected neq connection_rejected)   false;
        (connection_opened neq connection_opened)   false;
        (connection_closed neq connection_closed)   false;
        (init neq init)   false;
        (start neq start) = false;
        (stopp neq stopp) = false;
        ( alias_to_IP neq alias_to_IP)   false;
        (IP_to_alias neq IP_to_alias)   false;
        (response neq response)   false;
        (connection_req neq connection_req)   false;
        (C_C_exchange neq C_C_exchange) = false;
        (media_unit neq media_unit) = false;



ENDTYPE



(*************     Tipo de datos   PARAM            *************)

TYPE param IS Boolean

    SORTS
        param
    OPNS
        alias, IP      :-> param
        _eq_    :param, param  ->bool

    EQNS
    FORALL x,y:param
        OFSORT Bool

        (x eq x) = true;
        (x eq y) = false;
```

```
ENDTYPE

(*************************************************************************)
(**     COMPOSICION                                                     **)
(*************************************************************************)

BEHAVIOR
(((
        (((((Est_Ini_USR[CH_U_C_1] |[CH_U_C_1]|
        EST_Ini_Control[CH_U_C_1,      CH_C_MM_1,      CH_DB_C_1,      CH_C_H_1,      CH_MM_M_1,
CH_MM_H_1])|[CH_DB_C_1]|
        Est_Ini_DB[CH_DB_C_1])|[CH_C_H_1]|
        EST_Ini_H323[CH_C_H_1, CH_MM_H_1, CH_H_C0_1, CH_H_Co_1, CH_Co_N_1])|[CH_H_C0_1]|
        Est_Ini_Canal_0[CH_H_C0_1, CH_C0_N_1]) |[CH_C0_N_1, CH_Co_N_1]|
        Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2])
        |[CH_C0_N_2, CH_Co_N_2]|
        ((((Est_Ini_USR[CH_U_C_2] |[CH_U_C_2]|
        EST_Ini_Control[CH_U_C_2,      CH_C_MM_2,      CH_DB_C_2,      CH_C_H_2,      CH_MM_M_2,
CH_MM_H_2])|[CH_DB_C_2]|
        Est_Ini_DB[CH_DB_C_2])|[CH_C_H_2]|
        EST_Ini_H323[CH_C_H_2, CH_MM_H_2, CH_H_C0_2, CH_H_Co_2, CH_Co_N_2])|[CH_H_C0_2]|
        Est_Ini_Canal_0[CH_H_C0_2, CH_C0_N_2])
)|[CH_MM_M_1]|
Est_Ini_Media[CH_MM_M_1])     |[CH_MM_M_2]|
Est_Ini_Media[CH_MM_M_2])


WHERE



(*************************************************************************)
(**     CODIGO DE COMPORTAMIENTO                                        **)
(*************************************************************************)


PROCESS Est_Ini_DB[CH_DB_C]:NOEXIT:=

        CH_DB_C ? mensaje:msg;
        CH_DB_C ? parametro:param;
        (
                (
                        [mensaje eq alias_to_IP] -> (
                                CH_DB_C ! response;
                                CH_DB_C ! IP;
                                Est_Ini_DB[CH_DB_C]
                                )
                )
                []
                (
                        [mensaje eq IP_to_alias] -> (
                                CH_DB_C ! response;
                                CH_DB_C ! alias;
                                Est_Ini_DB[CH_DB_C]
                                )
                )
        )


ENDPROC



PROCESS Est_Ini_Media[CH_MM_M]:NOEXIT:=

        (CH_MM_M ? LDU:msg;
        Est_Ini_Media[CH_MM_M])
```

```
        []
        (CH_MM_M ! media_unit;
        Est_Ini_Media[CH_MM_M])

ENDPROC


PROCESS Est_Ini_Canal_0[CH_H_C0, CH_C0_N]:NOEXIT:=
(
        CH_H_C0 ? de_H323 :msg;
        (
                ([de_H323 eq connection_req] ->(
                        CH_H_C0 ? parametro :param;
                        CH_C0_N ! de_H323;
                        CH_C0_N ! parametro;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq connection_accepted] ->(
                        CH_C0_N ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq connection_rejected] ->(
                        CH_C0_N ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq C_C_exchange] ->(
                        CH_C0_N ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq close_connection] ->(
                        CH_C0_N ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )

        )

)
[]
(
        CH_C0_N ? de_H323 :msg;
        (
                ([de_H323 eq connection_req] ->(
                        CH_C0_N ? parametro :param;
                        CH_H_C0 ! de_H323;
                        CH_H_C0 ! parametro;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq connection_accepted] ->(
                        CH_H_C0 ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq connection_rejected] ->(
                        CH_H_C0 ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
```

```
                ([de_H323 eq C_C_exchange] ->(
                        CH_H_C0 ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )
                []
                ([de_H323 eq close_connection] ->(
                        CH_H_C0 ! de_H323;
                        Est_Ini_Canal_0[CH_H_C0, CH_C0_N]
                        )
                )

        )

)


ENDPROC


PROCESS Est_Ini_Co[CH_H_Co, CH_Co_N]:EXIT:=

        (
                CH_Co_N ? LDU:msg;
                CH_H_Co ! LDU;
                Est_Ini_Co[CH_H_Co, CH_Co_N]
        )
        []
        (
                CH_H_Co ? mensaje:msg;
                (
                        ([mensaje eq media_unit] ->
                                CH_Co_N ! mensaje;
                                Est_Ini_Co[CH_H_Co, CH_Co_N]
                        )
                        []
                        ([mensaje eq stopp] -> exit)
                )
        )

ENDPROC


PROCESS Est_Ini_MM[CH_MM_M, CH_MM_H, CH_C_MM]:EXIT:=

        (
                CH_MM_M ? LDU:msg;
                CH_MM_H ! LDU;
                Est_Ini_MM[CH_MM_M, CH_MM_H, CH_C_MM]
        )
        []
        (
                CH_MM_H ? LDU:msg;
                CH_MM_M ! LDU;
                Est_Ini_MM[CH_MM_M, CH_MM_H, CH_C_MM]
        )
        []
        (
                CH_C_MM ? mensaje:msg;
                ([mensaje eq stopp] -> exit)
        )

ENDPROC


PROCESS Est_Ini_USR[CH_U_C]:NOEXIT:=
```

```
        (
                CH_U_C ! open_connection;
                CH_U_C ! alias;
                Est_3_USR[CH_U_C]
        )
        []
        (
                CH_U_C ? mensaje :msg;
                CH_U_C ? parametro :param;
                (
                        (CH_U_C ! connection_rejected;
                        Est_Ini_USR[CH_U_C])
                        []
                        (CH_U_C ! connection_accepted;
                        Est_Ini_USR[CH_U_C])
                )
        )

ENDPROC


PROCESS Est_2_USR[CH_U_C]:NOEXIT:=

(
        CH_U_C ? mensaje :msg;
        (
                (
                        [mensaje eq connection_opened] ->(
                                Est_3_USR[CH_U_C]
                        )
                )
                []
                (
                        [mensaje eq connection_opened] ->(
                        Est_3_USR[CH_U_C]
                        )
                )
        )
)
[]
(
        CH_U_C ! close_connection;
        Est_Ini_USR[CH_U_C]
)

ENDPROC


PROCESS EST_3_USR[CH_U_C]:NOEXIT:=

        (
                CH_U_C ? mensaje:msg ;
                EST_Ini_USR[CH_U_C]
        )

ENDPROC


PROCESS EST_Ini_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]:NOEXIT:=

        (
                CH_C_H ? mensaje :msg;
                (
                        (
                                [mensaje eq connection_req] ->(
                                        CH_C_H ? parametro   param;
                                        CH_H_C0 ! mensaje;
                                        CH_H_C0 ! parametro;
                                        EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
```

```
                        )
                    )
                    []
                    (
                        [mensaje eq connection_rejected] ->(
                            CH_H_C0 ! mensaje;
                            EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje eq connection_accepted] ->(
                            CH_H_C0 ! mensaje;
                            EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje eq init] ->(
                            CH_H_C0 ! C_C_exchange;
                            EST_2_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                )
            )
            []
            (
                CH_H_C0 ? mensaje :msg;
                (
                    (
                        [mensaje eq connection_req] ->(
                            CH_H_C0 ? parametro   param;
                            CH_C_H ! mensaje;
                            CH_C_H ! parametro;
                            EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje eq connection_rejected] ->(
                            CH_C_H ! mensaje;
                            EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje eq connection_accepted] ->(
                            CH_C_H ! mensaje;
                            EST_Ini_H323[CH_C_H,    CH_MM_H,    CH_H_C0,    CH_H_Co,
CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje eq C_C_exchange] ->(
                            CH_H_C0 ! mensaje;
                            CH_C_H ! connection_ready;
                            EST_3_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                )
            )

ENDPROC
```

```
PROCESS EST_2_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]:NOEXIT:=

        CH_H_C0 ? mensaje:msg;
        (
                (
                        [mensaje eq C_C_exchange] ->(
                                CH_C_H ! connection_ready;
                                EST_3_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                )
                []
                (
                        [mensaje neq C_C_exchange] ->(
                                EST_2_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                )
        )

ENDPROC


PROCESS EST_3_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]:NOEXIT:=

        CH_C_H ?mensaje :msg;
        (
                (
                        [mensaje eq start] ->(
                                Est_Ini_Co[CH_H_Co, CH_Co_N] |[CH_H_Co, CH_Co_N]|
                                EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                )
                []
                (
                        [mensaje neq start] ->(
                                EST_3_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                )
        )

ENDPROC


PROCESS EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]:NOEXIT:=

        (
                CH_H_C0 ? mensaje :msg;
                (
                        (
                                [mensaje eq close_connection] ->(
                                        CH_C_H ! mensaje;
                                        EST_5_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                                )
                        )
                        []
                        (
                                [mensaje neq close_connection] ->(
                                        EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                                )
                        )
                )
        )
        []
        (
                CH_H_Co ? mensaje :msg;
                (
                        (
                                [mensaje eq media_unit] ->(
                                        CH_MM_H ! mensaje;
                                        EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
```

```
                        )
                    )
                    []
                    (
                        [mensaje neq media_unit] ->(
                            EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                )
            )
            []
            (
                CH_MM_H ? mensaje :msg;
                (
                    (
                        [mensaje eq media_unit] ->(
                            CH_H_Co ! mensaje;
                            EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje neq media_unit] ->(
                            EST_4_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                )
            )
            []
            (
                CH_C_H ? mensaje :msg;
                (
                    (
                        [mensaje eq close_connection] ->(
                            CH_H_C0 ! mensaje;
                            EST_5_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                    []
                    (
                        [mensaje neq close_connection] ->(
                            EST_5_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                        )
                    )
                )
            )

ENDPROC



PROCESS EST_5_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]:NOEXIT:=

            CH_C_H ? mensaje :msg;
            (
                (
                    [mensaje eq stopp] ->(
                        CH_H_C0 !mensaje;
                        EST_Ini_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                    )
                )
                []
                (
                    [mensaje neq stopp] ->(
                        EST_5_H323[CH_C_H, CH_MM_H, CH_H_C0, CH_H_Co, CH_Co_N]
                    )
                )
            )

ENDPROC
```

```
PROCESS EST_Ini_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        (
                CH_U_C ? mensaje :msg;
                (
                        (
                                [mensaje eq open_connection] ->(
                                        CH_U_C ? parametro :param;
                                        CH_DB_C ! alias_to_IP;
                                        CH_DB_C ! parametro;
                                        EST_2_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje neq open_connection] ->(
                                        EST_Ini_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )
        []
        (
                CH_C_H ? mensaje :msg;
                (
                        (
                                [mensaje eq connection_req] ->(
                                        CH_C_H ? parametro :param;
                                        CH_DB_C ! IP_to_alias;
                                        CH_DB_C ! parametro;
                                        EST_3_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje neq connection_req] ->(
                                        EST_Ini_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )

ENDPROC


PROCESS EST_2_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

       CH_DB_C ? mensaje :msg;
       (
               (
                       [mensaje eq response] ->(
                               CH_DB_C ? parametro :param;
                               CH_C_H ! connection_req;
                               CH_C_H ! parametro;
                               EST_4_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,   CH_MM_M,
CH_MM_H]
                       )
               )
               []
               (
                       [mensaje neq response] ->(
                               EST_2_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,   CH_MM_M,
CH_MM_H]
```

```
                        )
                )
        )

ENDPROC


PROCESS EST_3_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        CH_DB_C ? mensaje :msg;
        (
                (
                        [mensaje eq response] ->(
                                CH_DB_C ? parametro :param;
                                CH_U_C ! connection_req;
                                CH_U_C ! parametro;
                                EST_4_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,   CH_MM_M,
CH_MM_H]
                        )
                )
                []
                (
                        [mensaje neq response] ->(
                                EST_3_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,   CH_MM_M,
CH_MM_H]
                        )
                )
        )


ENDPROC


PROCESS EST_4_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        (
                CH_C_H ? mensaje :msg;
                (
                        (
                                [mensaje eq connection_accepted] ->(
                                        CH_C_H ! init;
                                        EST_5_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje eq connection_rejected] ->(
                                        CH_U_C ! connection_closed;
                                        EST_Ini_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )
        []
        (
                CH_U_C ? mensaje :msg;
                (
                        (
                                [mensaje eq connection_accepted] ->(
                                        CH_C_H ! mensaje;
                                        EST_5_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje eq connection_rejected] ->(
                                        CH_C_H ! mensaje;(**close_connection;**)
```

```
                                        EST_Ini_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )

ENDPROC


PROCESS EST_5_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        CH_C_H ? mensaje :msg;
        (
                (
                        [mensaje eq connection_ready] ->(
                                CH_C_H ! start;
                                CH_U_C ! connection_opened;
                                (
                                        Est_Ini_MM[CH_MM_M,    CH_MM_H,    CH_C_MM]    |[CH_MM_M,
CH_MM_H, CH_C_MM]|
                                        EST_6_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
                []
                (
                        [mensaje neq connection_ready] ->(
                                EST_5_Control[CH_U_C,   CH_C_MM,   CH_DB_C,   CH_C_H,   CH_MM_M,
CH_MM_H]
                        )
                )
        )

ENDPROC


PROCESS EST_6_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        (
                CH_C_H ? mensaje :msg;
                (
                        (
                                [mensaje eq close_connection] ->(
                                        CH_U_C ! connection_closed;
                                        CH_C_MM ! stopp;
                                        CH_C_H ! stopp;
                                        EST_Ini_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje neq close_connection] ->(
                                        EST_6_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )
        []
        (
                CH_U_C ? mensaje :msg;
                (
                        (
                                [mensaje eq close_connection] ->(
                                        CH_U_C ! connection_closed;
                                        CH_C_MM ! stopp;
                                        CH_C_H ! close_connection;
```

```
                                        EST_7_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                        []
                        (
                                [mensaje neq close_connection] ->(
                                        EST_6_Control[CH_U_C,    CH_C_MM,    CH_DB_C,    CH_C_H,
CH_MM_M, CH_MM_H]
                                )
                        )
                )
        )

ENDPROC


PROCESS EST_7_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]:NOEXIT:=

        CH_C_H ! stopp;
        EST_Ini_Control[CH_U_C, CH_C_MM, CH_DB_C, CH_C_H, CH_MM_M, CH_MM_H]

ENDPROC


PROCESS Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2]:NOEXIT :=

        (
                CH_C0_N_1 ? mensaje :msg;
                CH_C0_N_2 ! mensaje;
                (
                        (
                                [mensaje eq connection_req] ->(
                                        CH_C0_N_1 ? parametro :param;
                                        CH_C0_N_2 ! parametro;
                                        Est_Ini_Network[CH_C0_N_1,    CH_C0_N_2,    CH_Co_N_1,
CH_Co_N_2]
                                )
                        )
                        []
                        (
                                Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2]
                        )
                )
        )
        []
        (
                CH_C0_N_2 ? mensaje :msg;
                CH_C0_N_1 ! mensaje;
                (
                        (
                                [mensaje eq connection_req] ->(
                                        CH_C0_N_2 ? parametro :param;
                                        CH_C0_N_1 ! parametro;
                                        Est_Ini_Network[CH_C0_N_1,    CH_C0_N_2,    CH_Co_N_1,
CH_Co_N_2]
                                )
                        )
                        []
                        (
                                Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2]
                        )
                )
        )
        []
        (
                CH_Co_N_1 ? mensaje :msg;
                CH_Co_N_2 ! mensaje;
                (
```

```
                        (
                                [mensaje eq connection_req] ->(
                                        CH_Co_N_1 ? parametro :param;
                                        CH_Co_N_2 ! parametro;
                                        Est_Ini_Network[CH_C0_N_1,     CH_C0_N_2,     CH_Co_N_1,
CH_Co_N_2]
                                )
                        )
                        []
                        (
                                Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2]
                        )
                )
        )
        []
        (
                CH_Co_N_2 ? mensaje :msg;
                CH_Co_N_1 ! mensaje;
                (
                        (
                                [mensaje eq connection_req] ->(
                                        CH_Co_N_2 ? parametro :param;
                                        CH_Co_N_1 ! parametro;
                                        Est_Ini_Network[CH_C0_N_1,     CH_C0_N_2,     CH_Co_N_1,
CH_Co_N_2]
                                )
                        )
                        []
                        (
                                Est_Ini_Network[CH_C0_N_1, CH_C0_N_2, CH_Co_N_1, CH_Co_N_2]
                        )
                )
        )


ENDPROC


ENDSPEC
```

Apéndice C

Voice Over IP
Software Design Specification

Originator: David Garduño & Ivan Romero
Date: January 28, 2000
Issue: 0.6

Software Requirements Specification for the System VoIP

1 Introduction

This document specifies the User Requirements for the VoIP system. It encompasses the global requirements for the VoIP and the intended interfaces between hardware and software components of the system.

The goal of the VoIP system is to establish a common communication framework for office environments. This framework is based on the LAN environment commonly found on most organizations. It is intended to facilitate the migration from the current telephony equipment to a fully computer based communications system. The VoIP is oriented to achieve fully connectivity with equipment based on H323 standard

The worldwide market for IP telephony equipment alone could reach $3.2 billion by 2002. [6] This product is being developed in order to reduce costs of long distance telephone calls and taking advantage of the actual global computer network and cost reduction. It has been estimated that Internet telephony will be a US $560 million market by the end of 1999. By the end of 1997, IP telephony traffic reached 6.3 million minutes per month, according to a report released in May 1998 by marketing consultants Frost & Sullivan. That represents just 0.2 percent of all telephone traffic, but the report predicts an increase to 10 percent of all traffic by 2002.
Notes: When using the word ***shall***, we mean obligation, end when using the word ***might***, we mean option.

1.1 Purpose

This document is written for designers, test engineers and people interested in the subject. This document will be updated anytime the product changes.

This document shall set the basis for a full multimedia communications implementation, however, here we specify voice communications only.

1.2 Definitions, Acronyms and Abbreviations

Host	Whichever terminal involved in a communication
IP	Internet Protocol
ITU-T	International Telecommunications Union- Telecommunications
LAN	Local Area Network
Partner	Whichever terminal involved in a communication
PHNTS	Phone number to the network address Translation Service
PSTN	Public Switched Telephone Network
QoS	Quality of Service
Sys Admin	Privileged user, it manages and configures the VoIP system
Terminal	Can be a Gatekeeper, Gateway, compressing machine or communicating machine. The computer containing IP/6 Port FXS cad and the one containing IP/4 Trunk card are terminals too.
VoIP	Voice over IP
WAN	Wide Area Network

1.3 References

[1] ANSI/IEEE Std. 830-1984, IEEE Guide to Software Requirements Specification.
[2] IP/4 Port Trunk card technical design specification
[3] IP/ G.723.1 compression card technical design specification

[4] IP FXS card technical design specification (6port)
[5] *http://computer.org/internet/telephony/service.htm.* "IEEE Internet Computing on Line" 1999. Institute of Electrical and Electronics Engineers, Inc.
[6] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.323, Packet-Based Multimedia Communications Systems» 1996
[7] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.225.0, Media Stream Packetization and Synchronization for Visual Telephone Systems on non-guaranteed Quality of Service LAN's» 1998
[8] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.245, Control Protocol for multimedia communications» 1998
[9] CCITT «Rec. G.711, Pulse Code Modulation (PCM) of voice frequencies» 1988
[10] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. G.723.1, Speech Coders: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbps» 1996
[11] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.261, Video Codec for Audiovisual Services at p X 64 kbps» 1993
[12] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. T.120, Transmission Protocols for Multimedia Data» 1996
[13] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.320, Narrow-Band visual Telephone Systems and Terminal Equipment» 1996
[14] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.324, Terminal for Low Bit Rate multimedia communications» 1996
INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. Q.931, ISDN User-Network Interface Layer 3 Specification for Basic Call Services» 1996

1.4 Overview

The system is called VoIP (Voice over Internet Protocol). The concept is to provide the telephone & multimedia service over IP networks. The system uses the resources already existing in an organization (multimedia PCs, classic telephones sets, fax machines, etc.), see Fig 1.

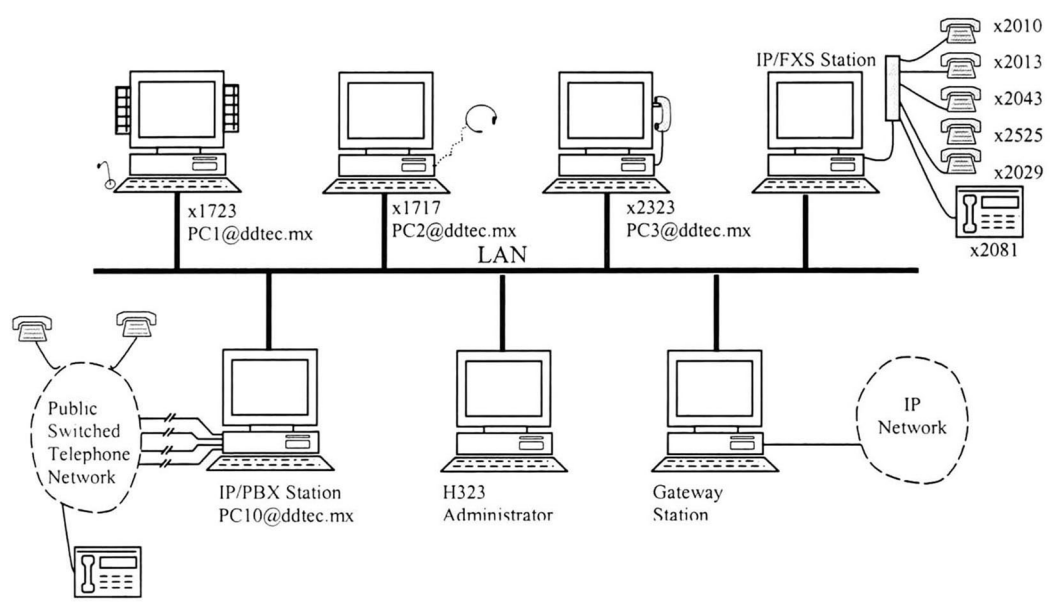


Figure 1. VoIP system

The VoIP software follows the H323 recommendation and it is the control component of the whole system.

VoIP software shall do:

- Interact with specialized hardware for the PSTN and telephone sets and compression.
- Integrate algorithms for interflow and intraflow Multimedia Synchronization.
- Transmit multimedia data through LAN or WAN via IP.
- Comply with ITU-H.323 standards.
- Telephone Service via IP networks
- Call establishment
- Call in progress monitoring
- Call end
- Special services (waiting, bypass, do not disturb, voice mail)

VoIP will not:

- Ensure QoS.
- Ensure the establishment of a call (case of network failure).

1.5 Scope

This document is organized as follows: section 2, general description, describes general factors those affect the product and its requirements; section 3, specific requirements, contains the details of the system being specified; appendixes are given in last section.

2 General Description

The VoIP project is a Computer based Communication System for office environments. It is an extension to actual telephone services with a clear evolution towards videoconference services integrating management and accounting features.

2.1 Product Perspective

The system comprises the following hardware:

- IP FXS card (6port). Interface between 6 telephone sets and the PCI Bus of a PC.
- IP/4 Port Trunk card. Interface between 4 PSTN trunks and the PCI Bus of a PC.
- IP/ G.723.1 compression PCI based card.

The glue of these components is the H.323 compliant software control system. It is in charge of the coordination between components of the system, management, Billing, security, consistency, and telephony related services.

2. 2 VoIP Interfaces

The system environment is depicted in fig. 2. It is network topology independent, based on LAN installation with the requirement of using IP as networking protocol.

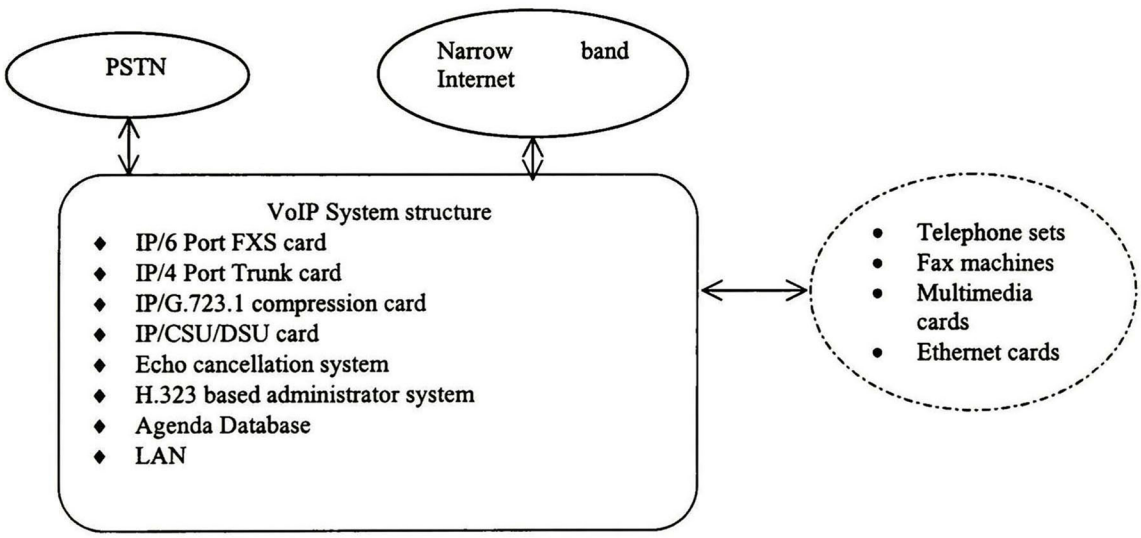


Figure 2. VoIP interfaces

2.2.0 Device Drivers Interfaces

The central software or control software shall not interact directly with hardware. Strictly, it shall not interact with IP FXS, IP Trunk, IP/CSU_DSU and ITU G.723.1 cards but with its Device Drivers.

All along this document, when saying that Control Software *shall send to or receive from* any of above cards, we mean, *send to or receive from* its corresponding Device Driver.

So, any one of the above cards shall interact with its Device Driver, the Device Driver shall interact with Control Software and Control software interacts with Device Drivers and IP Network

For details on this communications, see chapter 7

2.2.1 IP / 4 Port Trunk card

The trunk card is responsible of connecting the local area network to the telephone central office. The card supports up to 4 trunks. The interface to the host computer is made through the Bus PCI running at 33 MHz. For more information on messages interchange, see chapter 7.

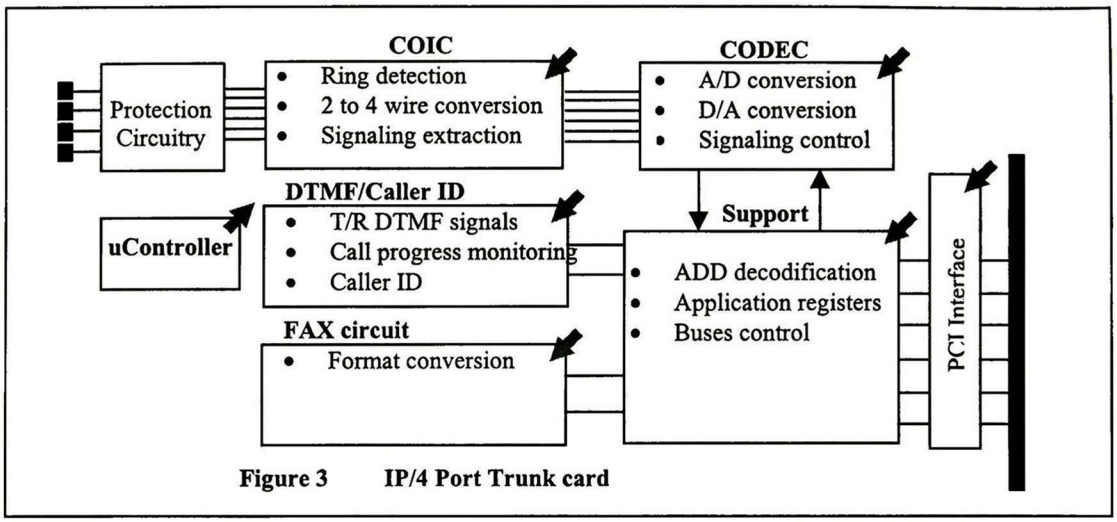
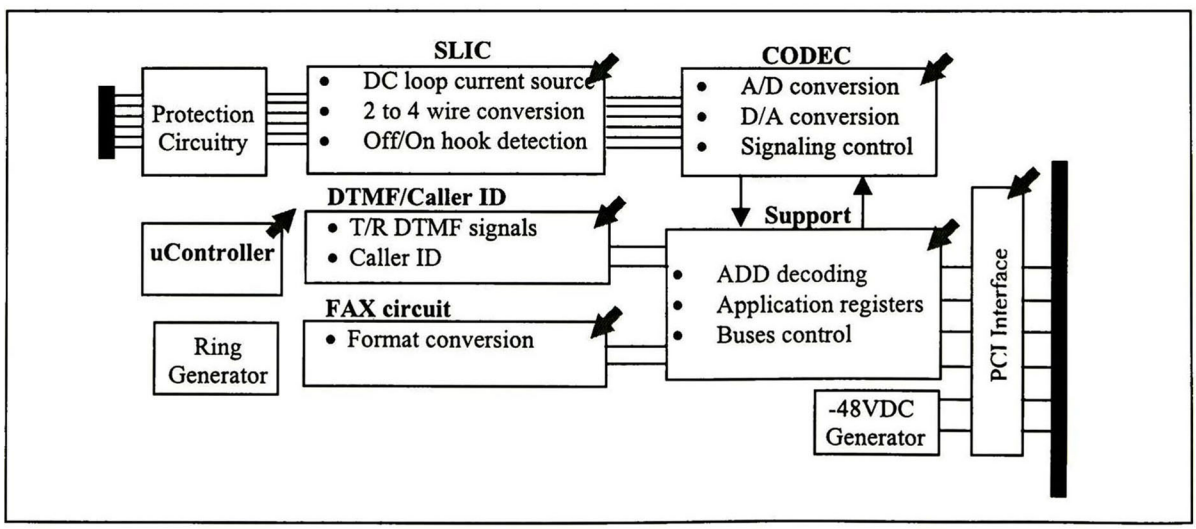


Figure 3 IP/4 Port Trunk card

2.2.2 IP FXS card (6 port)

The FXS card is responsible of connecting the traditional telephone sets to the local area network. The card supports up to 6 telephone sets. The interface to the host computer is made through the Bus PCI running at 33 MHz.

For more information on messages interchange, see chapter 7.



2.2.3 IP / ITU G.723.1 compression card

The compression card is responsible of compressing and decompressing the voice information to be transmitted to or received from the router to Internet. The card is based in a DSP to carryout these tasks. The card supports up to 10 simultaneous phone calls. The interface to the host computer is made through the Bus PCI running at 33 MHz.

For more information on messages interchange, see chapter 7.

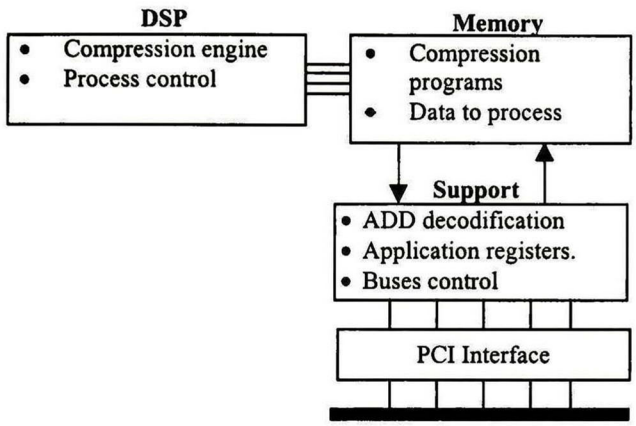


Figure 5. ITU G.723.1 compression card

2.2.4 IP / CSU_DSU card

The CSU_DSU card is responsible of connecting the local area network to a T1/E1 digital line. The system supports T1/E1 or FT1/FE1 formats. Up to 24/32 calls can be handled by the card. Data and voice channels can be managed simultaneously. The interface to the host computer is made through the Bus PCI running at 33 MHz.

For more information on messages interchange, see chapter 7.

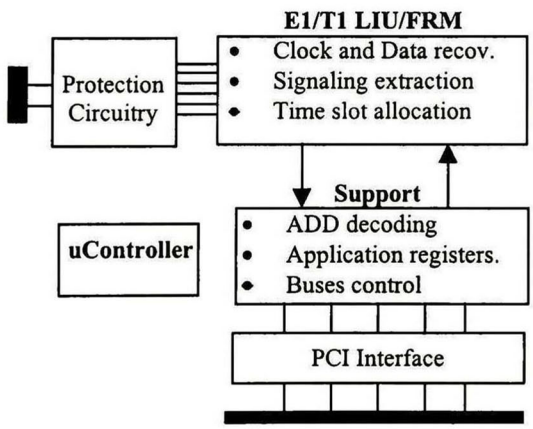


Figure 6. CSU_DSU card

2.2.5 H323 administrator

The H.323 administrator is a core of programs, which make possible to implement the phone service over the data network. The administrator is ITU-T H.323 specification compliant:
This functions shall be done in Gatekeeper, which is a central element in H.323 (see chapter 5)

- **User definition**
  - User account creation or deletion
  - Modification of attributes and privileges to the users
  - Foreign users access control

- **Call establishment**
  - Operates the following telephone services: intraoffice, interoffice, Traditional phone service, Bypass see fig 7, 8, 9 and 10
  - Negotiation of the resources to be utilized by the user
  - Control of the dialing and signaling
  - Phone number mapping to their equivalent IP address and viceversa
  - Greetings service
  - Caller identification
  - Automatic switch to an extension number
  - Conference set-up
  - Synchronize multiple communications from the same source to the same destination
  - Telephone numbers and services restrictions

- **Call in progress monitoring**
  - Status of the call
  - Record of the user calls and statistics (number called, duration, and date)
  - Billing service

- **Call end**
  - Release of the resources utilized by a user

- **Special services**
  - Telephone directory
  - Agenda Database
  - Voice mail stored at end terminal
  - Security services

2.2.6 Agenda database

Upon the identification of the user to be called or the identification of the phone number of the incoming call, the system shall display in the screen the information available about the person calling or to be called.

The database should follow the following rules:

- Adapted to the application (Medical care center, Banks, Travel agencies, Government agencies, etc.)
- The data to be displayed can follow any customer-preferred format (from a format set)
- Data can be created, edited and updated during the call
- Data is stored in local database
- Data can be updated from/to general database

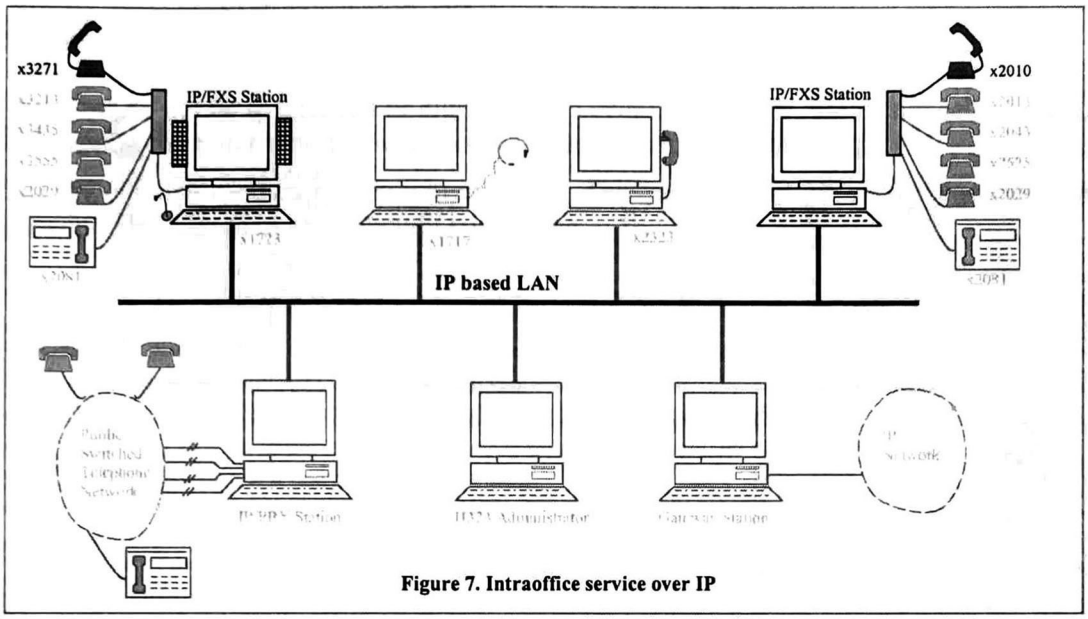


Figure 7. Intraoffice service over IP

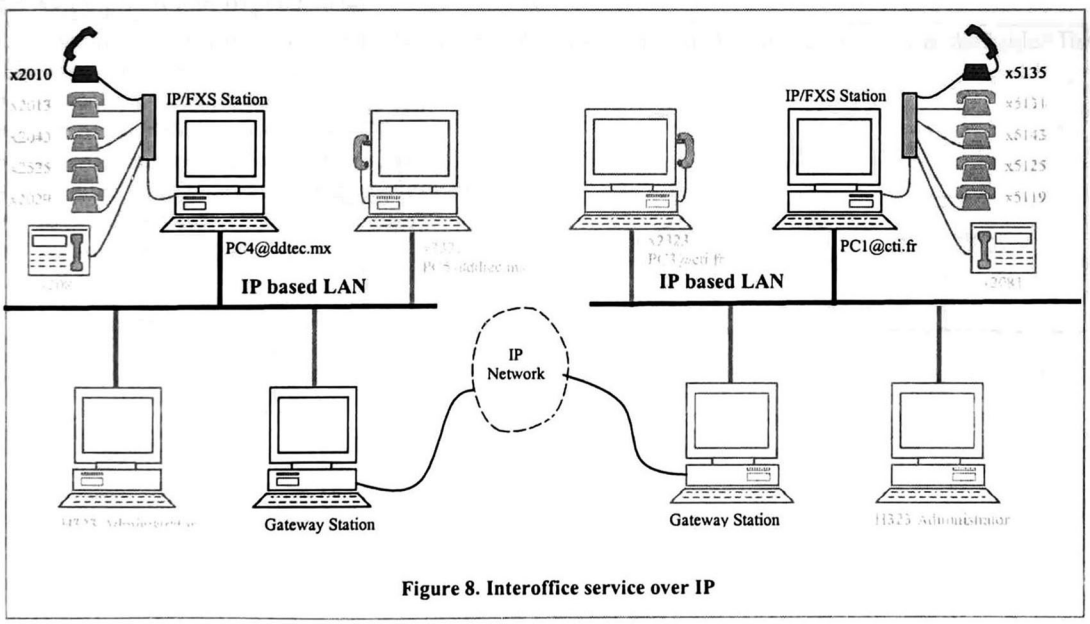


Figure 8. Interoffice service over IP

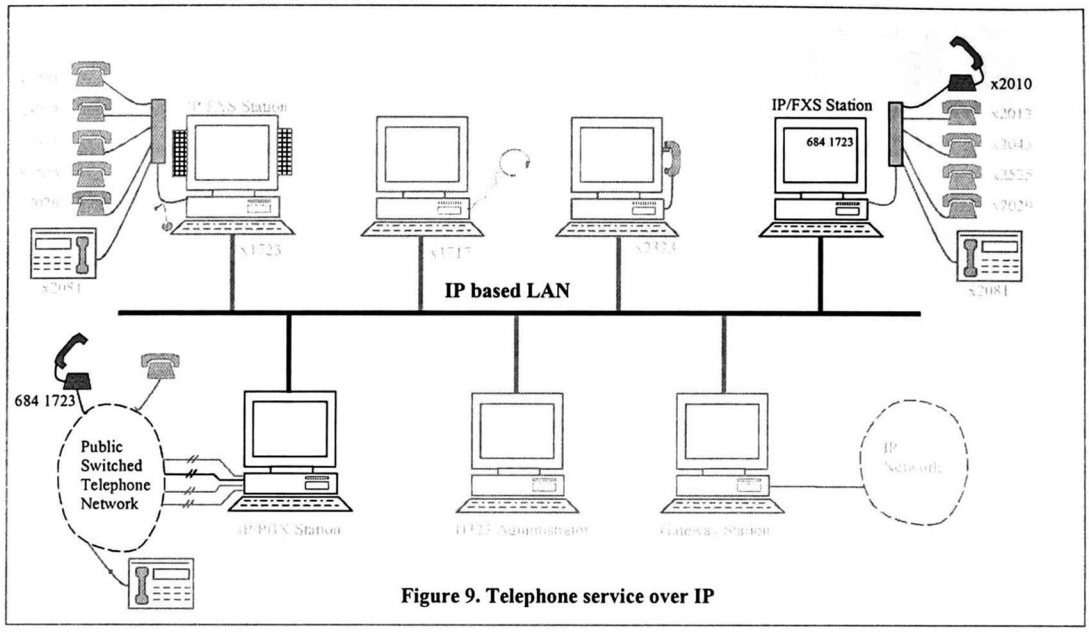


Figure 9. Telephone service over IP

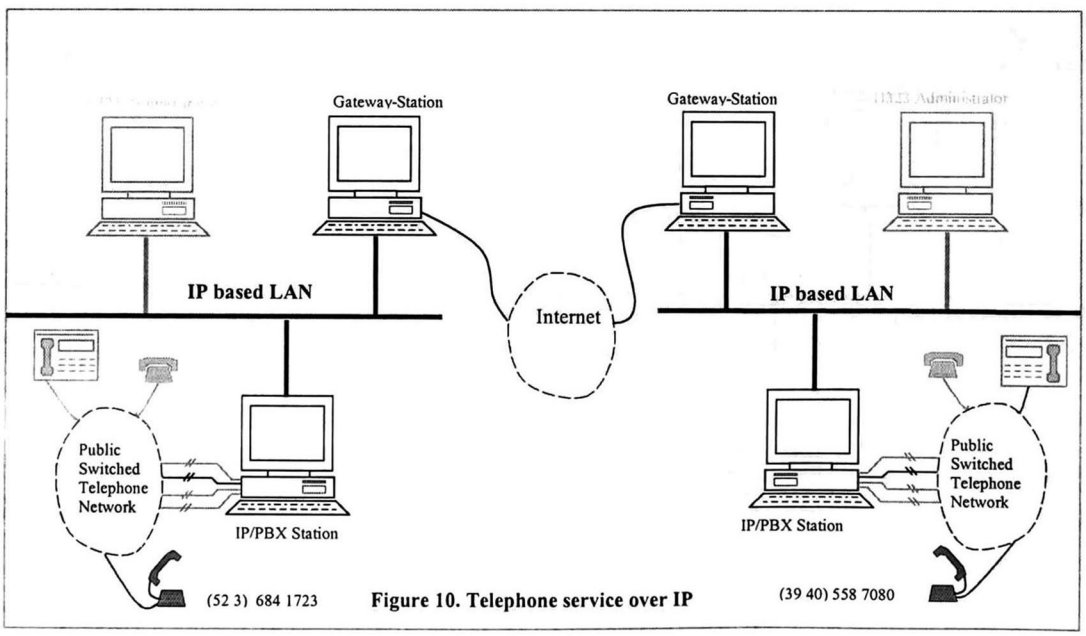


Figure 10. Telephone service over IP

2.3 User Characteristics

The users do not be specialized or highly skilled people. However, users connected through a dedicated computer terminal will need a minimal computing knowledge.

The system also needs a system administrator. This person shall have deep knowledge in network administration and database knowledge.

2.4 General Constraints

The system will be limited by the following policies:

- It shall compliant with ITU-T H.323 Specifications and associated standards.
- It shall work over an IP network, making transparent the subjacent network.
- It shall synchronize and deliver multimedia data.
- Provisions to handle three security levels have to be taken:
  Accounting level: to provide a policy based on costs.
  Management level: to provide geographical restriction, in both ways to avoid be reached or to reach a defined site.
  Security level: to provide a compression and encryption service.
  It is possible to have a mixed operation of these three levels.
- It has to have an interface with PSTN.
- It can be accessed through dedicated H323 equipment.

2.5 Assumptions and Dependencies

It is assumed the existence of the LAN and PSTN connections. An Internet connection is desirable. The following equipment is required:

- A network interface board
- An IP network connection
- Microphone and loud speakers (optional)
- Pentium equivalent microprocessor (or better)
- RAM Memory of 32 MB as minimum.
- Main storage: 64 MB.

2.6 Future Perspectives

The VoIP will evolve in the following three stages.

d) In the first phase only a LAN environment is considered. Only the H323 related to communications on a local environment are considered. The communication is established using IP, making independent the system of any particular topology.
e) In the 2nd phase interconnection with Internet will be the main concern. This interconnectivity will involve security aspects, multimedia synchronization, QoS, interconnection with PSTN.
f) In the 3rd phase multicast IP videoconferences with initial document handling (log, conference recording,...) shall be implemented.

3. Requirements

3.1 General

R.1.1 Phone services

- Intraoffice (from an extension number to another extension number in the same IP Network)
- Interoffice (from an extension number to another extension number in a different IP Network)
- Local or long distance service through the PSTN
- Long distance service through the IP network (Bypass included)

R.1.2. Call destination/origin

- To/from the IP Network (multimedia H323 based PC terminal, telephone set or Fax. Machine)
- To/from the PSTN (telephone set or Fax. machine)

R.1.3. Valid phone numeration

The system shall accept and process the following destination numbers

- PSTN-like numbers
  Extension numbers (example: 2010, 2080, etc.)
  Traditional phone numbers (long distance access code + country code + area code + phone number)
- IP-like numbers
  Alphanumeric strings (example: juans@ddtec.com.mx, 345.2@141.208.208.3)

R.1.4. Address mapping

- Inside the IP Network, call destinations can be handled as IP numbers or PSTN alike.
- PSTN-like numbers (dialed from the inside the IP Network) shall be mapped to IP-like numbers by the administrator for processing the call

R.1.5. Signaling

The system protocol shall support signaling for the following events:

- Invitation to dial tone
- Calling tone
- Busy tone
- Dialed number
- Ringing
- Call waiting
- Caller ID

R.1.6. Security

- The system protocol shall support voice encryption

3.2 Establishing a call from a PC

R.2.1. Dialing alternatives from a PC

- From the keyboard
- From a personal agenda (file in the host computer)
- From a corporation directory (available on the network, file owned and controlled by the H323 system and located at Gatekeeper)

R.2.2. Inputs required to establish the call from a PC

- Phone number from keyboard
- Call acceptance from the user (from keyboard)
- Call established signal from the network
- Disconnection request from the user or from the network
- Connection refused signal from the network

R.2.3 Processing required to establish the call from a PC

- Open reliable communication channels for control data transference (H.245)(open connection, close connection, accept connection, refuse connection)
- Open not reliable communication channel for voice transference (RTP)
- Calling terminal sends ID information to the called partner.
- Transferring control data to/from partner (disconnection request, reject connection, etc.)
- Matching agenda entrance to IP or viceversa (if required)
- Generate ring back signal to speaker and graphic signal to the screen
- Notify the user about connection accepted or rejected event
- Send call signaling to speaker and screen (busy, calling, invitation)

R.2.4 Outputs expected

- ID to the network for caller ID. This because it's necessary to establish a network connection before telephone connection. Also, all control data shall be sent via reliable channel.
- Ring back signal to the speaker and graphic signal to screen
- Connection request to the network (to the called partner)
- Disconnection request to the network
- Busy signal to speaker
- Disconnection request to user (coming from called partner)

R.2.5 Performance requirements

The connection and disconnection processes have no time restrictions, but voice transmission does.

R.2.6 Design constraints

While transmitting and receiving voice data to and from the network, it shall compliance with ITU – T H.323 recommendations, as in connection, disconnection and agreement processes. Next chronogram shows dialing from any terminal as extracted from H.323 specification.

In this scheme appear messages belonging to three different protocols : RAS ,H.225 and H.245. Every protocol has its own specification from ITU and messages are clearly and completely defined. H.323 uses a mixture of protocols interacting strongly inside the telephony system entities. H.323 specifies many parts of the system in a very clear manner, some others are intentionally left to the implementers as Multimedia synchronization , presentation and accounting policies.

When calling from a PC, it shall compliance with audio input and output standard interfaces.

R.2.7 Other requirements

The PC terminal system, shall be capable for using an agenda for matching PSTN numbers to IP numbers, this agenda shall comply with standard SQL databases.
Other database requirements are implemented in section 3.6, billing.

3.3 Establishing a call from a telephone set connected to an IP/FXS card

R.3.1 Dialing alternatives from a telephone set connected to an IP/FXS card

- From telephone keyboard

R.3.2 Inputs required to establish the call from a telephone set connected to an IP/FXS card

- The number to call
- Or keyword for calling (password, if required)
- Or extension number to call
- Call established signal from the network
- Connection refused signal from the network
- Voice data from a telephone set connected to an IP/FXS card (read trough Device Driver)
- Voice data from the network
- Disconnection request from network
- Hang on signal from a telephone set connected to an IP/FXS card (read trough Device Driver)
- Hang off signal from a telephone set connected to an IP/FXS card (read trough Device Driver)

R.3.3 Processing required to establish the call from a telephone set connected to an IP/FXS card

- Open reliable communication channels for control transference (H.245)
- Open not reliable communication channel for voice transference (RTP)
- Sending ID information to the partner
- Transferring control data to/from partner (disconnection request, reject connection, etc.)
- Send call signaling (invitation, busy, ring back) to the telephone set connected to an IP/FXS card (read trough Device Driver). Control software sends a *start ring back* signal to the IP/FXS card, and the IP/FXS card keeps sending this signal to the telephone set till receiving a *hang on* signal from telephone set or receiving *stop ring back* signal from Control Software.
  Something similar happens with other signals in this and the other cards.
- Sending call signaling to telephone set (invitation, busy, calling)

R.3.4 Outputs expected

- Connection request to the network
- Voice data to the network
- Voice data to a telephone set connected to an IP/FXS card (read trough Device Driver) through its Device Driver
- Ring-back signal to the telephone set connected to an IP/FXS card (read trough Device Driver) through its Device Driver
- Busy signal to the telephone set connected to an IP/FXS card (read trough Device Driver) through its Device Driver
- Dial signal to the telephone set connected to an IP/FXS card (read trough Device Driver) through its Device Driver

R.3.5 Performance requirements

The system shall emulate a standard telephone behavior. The connection and disconnection processes have no time restrictions.

R.3.6 Design constraints

The system shall comply with standard telephone standards in the telephone side and with H.323 in the network side.

The system shall interact with IP/6 port FXS (telephone) card.

3.4 Call from a telephone set connected to the PSTN.

The VoIP has the capability of processing a call originated from a telephone set attached to a Public Switched Telephone Network to an inside system terminal and viceversa. There is no special user skills are required. A switch system may answer the user call from outside the system, this automated system will ask for the desired extension number and will connect this external communication to the internal terminal required.

R.4.1 Dialing alternatives from a telephone set connected to the PSTN

- From telephone keyboard

R.4.2 Inputs required to establish the call from a telephone set connected to the PSTN

- A notification of trunk ringing
- The telephone extension number to be communicated with (from IP/Trunk card)
- The number to be communicated with (from IP/Trunk card)
- Validation keyword (if required) (from IP/Trunk card)
- The connection accepted signal from the network
- Voice data from the network
- Voice data from from IP/Trunk card
- Connection refused from the network
- Disconnection signal from the network
- Busy signal from IP/Trunk card
- DTMF signal from IP/Trunk card

R.4.3 Processing

- Open reliable communication channels for control transference (h.245)
- Open not reliable communication channel for voice transference (RTP)
- Send ID information to the called partner
- Transferring control data to/from partner (disconnection request, refuse connection, etc.)
- When the IP/Trunk card sends DTMF, the control software shall differentiate ID keyword from extension number. If the received DTMFs are ID keyword, the control software shall validate with gatekeeper, if it's invalid, control software shall send a new locution to the IP/Trunk card, otherwise shall continue with service. All DTMFs received in the IP/Trunk card and the IP/FXS card, shall be sent trough reliable channels.

R.4.4 Outputs.

- Start Ring back signal to the IP/Trunk card
- Stop ring back signal to the IP/Trunk card
- Connection request signal to the network
- Voice data to the network
- Voice data to the IP/Trunk card
- Busy signal to the IP/Trunk card
- Disconnection signal to the network
- On hook signal to the IP/Trunk card
- Locution to the IP/Trunk card

R.4.5 Design constraints

The system shall interact with IP/4 Trunk card

3.5 Digital Services.

Digital services are desired features in telephone systems. VoIP shall implement some of them, but not all. However, it shall open the doors for future improvements. The way of doing it is by using a flexible architecture. The number of services and the way they should be implemented, are not explained here.

R.5.1 Inputs required to establish digital services

- The prefix for digital services

R.5.2 Processing required to establish digital services

When receiving the prefix for digital service, the system shall output the service request for the network.

R.5.3 Expected outputs.

- Service request for the network

R.5.4 Design constraints

There are no standards for telephone digital services, so the system shall implement digital services by copying or imitating established digital services in foreign systems, but documenting them for future improvements.

The hardware limitations depend on the terminal from which the call is done.

R.5.5 Attributes

The system shall implement security access procedures for the company databases. Access shall be forbidden to non-authorized users calling from a telephone set connected to the PSTN.

3.6 Receiving a call in a PC

An arriving call can be taken in different voice terminals that can reproduce the audio. The one of them is explained below. Somebody is calling from anywhere and the call is received in a PC terminal using a graphic interface to let the user to interact with the system interacting with user. And microphone and loudspeakers are used for receiving and reproducing the audio.

R.6.1 Inputs required for receiving a call in a PC

- Connection request from the network
- Voice data from the network and microphone
- Call acceptance from the user (keyboard)
- Disconnection request from the network
- Disconnection request from the user (keyboard)

R.6.2 Processing required to receive a call in a PC

- When receiving a connection request from the network, the control software shall send a ring signal to the loudspeakers and a message or graphics to the display
- Wait for an answer from the user (accept/reject) and notify it to the network
- Notify the user when connection is established
- Notify the user when connection is finished
- Map IP addresses into their corresponding aliases
- Transferring control data to/from calling partner
- Show calling partner's alias on screen

R.6.3 Expected outputs

- Voice data to the network and the loudspeaker
- Ring signal to the loudspeaker
- Call attempting message to the screen
- Connection accepted to the network
- Connection refused to the network
- Disconnection request to the network
- Busy signal to the loudspeaker

R.6.4 Design constraints

The system shall interact with standard input-output audio interfaces for PC's (microphone and loudspeakers).

3.7 Receiving a call in a telephone set connected to the IP/FXS card

Several telephone sets may be connected to a PC through FXS card. The explanation below refers to the process followed to accept a call in a telephone set.

R.7.1 Inputs required for receiving a call in a telephone set connected to the IP/FXS card

- Connection request from the network
- Hang off signal from the IP/FXS card
- Voice data from the network and the IP/FXS card
- Hang on signal from the IP/FXS card
- Disconnection request from the network
- ID information from the network
- DTMFs from the IP/FXS card
- The number or the extension number to call

R.7.2 Processing required for receiving a call in a telephone set connected to the IP/FXS card

- Open reliable communication channels for control data transference (H.245)
- Open not reliable communication channel for voice data transference (RTP)
- Transferring control data to/from partner (disconnection request, reject connection, etc.)
- Receiving ID data from the network and send it to the IP/FXS card
- Sending call signaling to the IP/FXS card (invitation, busy, calling)
- When receiving a connection request from the network, the control software shall send a Start Ring signal to the IP/FXS card and wait till the IP/FXS card sends a hang off signal or receiving a disconnection request from the network.
- When IP/FXS card send the hang off signal, the control software shall send a Stop Ring signal to the IP/FXS card
- When the IP/FXS card sends a hang on signal to the control software, it shall send a disconnection request to the network
- When receiving a disconnection request from the network, the control software shall send a Start Busy signal to the IP/FXS card and wait till receive a Hang on signal from the IP/FXS card and send a Stop Busy signal to the IP/FXS card

R.7.3 Expected outputs

- Start Ring signal to the IP/FXS card
- Stop Ring signal to the IP/FXS card
- Connection accepted signal to the network
- Close connection signal to the network
- Start Busy signal to the IP/FXS card
- Stop Busy signal to the IP/FXS card
- Voice data to the network and to the IP/FXS card
- DTMFs to the IP/FXS card (if required)

R.7.4 Design constraints

The system shall interact with IP/6 port FXS card.

3.8 Receiving call in a telephone set connected to the PSTN

When calling from inside the system, the called terminal could be an external one. So there shall be an internal terminal for interfacing with PSTN. For calling an external terminal the user shall first contact with this interfacing terminal and after this with the external terminal. The interface terminal behavior is explained below.

R.8.1 Inputs required for receiving call in a telephone set connected to the PSTN

- Connection request from the network
- User ID from the network
- External telephone number from the network
- Voice data from the network and the IP/Trunk card
- Disconnection request from the network
- Busy signal from the IP/Trunk card
- DTMFs from the IP/Trunk card

R.8.2 Processing required for receiving call in a telephone set connected to the PSTN

- Open reliable communication channels for control transference (H.245)
- Open not reliable communication channel for voice transference (RTP)
- Sending caller ID information to the IP/Trunk card
- Transferring control data to/from partner (disconnection request, reject connection, etc.)
- Sending call signaling to IP/Trunk card (invitation, busy, calling)
- The control software shall perform billing for all outgoing calls

R.8.3 Expected outputs

- Disconnection request signal to the network
- Voice data to the network and the IP/Trunk card
- Hang on signal to the IP/Trunk card
- Hang off signal to the IP/Trunk card
- Connection accepted to the network
- Connection refused to the network
- DTMFs to the network
- DTMFs to the IP/Trunk card

R.8.4 Design constraints

It is necessary to comply with times in standard telephone communications in PSTN side.
In network side it is necessary to comply with ITU - T H.323 standards.

The system shall interface with IP/Trunk card.

R.8.5 Attributes

There must exist keywords or passwords for calling to an external terminal and all the data (ID, external number, time and date) must be recorded.

R.8.6 Other requirements

The system shall record all the calls from/to outside

The PSTN interface terminal shall be adapted as a gateway in order to interact with PSTN.

3.9 Digital Services.

Digital services when receiving a telephone call are implemented just as if the call would have been generated in this terminal. So, the explanation is just the same as in digital services for making a call, section 3.1.4.

3.10 Ending a call.

When calling or receiving a call from a telephone set connected to the PSTN, in the moment of ending a call, the terminal shall send an end session message to the partner. The whole data referred to the call shall be recorded. Also, the terminal shall notify the gatekeeper (if any) for releasing channels.

3.11 System administration: Extension-IP translation

Given that the VOIP system needs to translate extension numbers to the network addresses, there shall be a phone number to the network address translation service (PHNTS). That subsystem provides the necessary function using tables relating numbers and network addresses. Any terminal establishing a call asks this subsystem for retrieving the network address holding the given phone number.

In fact this subsystem manages the extension assignment in the whole system, it is required that every terminal in the system must be registered in the PHNTS for being able to make and receive calls using extension numbers.

R.11.1 Inputs required for Extension-IP translation

- Add terminal phone and network address (number, network address, description, permissions) from console
- Release terminal by phone or by address, from console.
- Terminal number or network address to edit, from console
- Extension or phone number to translate, from the network

R.11.2 Processing required for Extension-IP translation

- Capturing new Extension data
- Mapping IP into Extension data and viceversa
- Receiving new data from gatekeeper

R.11.3 Expected outputs

- Network address associated with phone number.
- Confirmation message

R.11.4 Performance requirements

Changes done must be immediately available to any terminal establishing a new phone call. The changes don't affect currently established calls. The system has not other performance requirements.

R.11.5 Design constraints

The subsystem must be H.323 compliant, so the communication shall be implemented using the protocols specified in the ITU recommendation and the final implementation must include all the basic services suggested. Any extra service not specified by ITU may be included in the system,

because H.323 specification allows inclusion of non-standard services. The subsystem shall be represented as a H.323 Gatekeeper with added functionality.

The subsystem has not hardware limitations, because it does not interact with any specific hardware beyond the necessary for being executed in its OS and for interacting with TCP/IP protocols stack.

R.11.6 Attributes for Extension-IP translation

The subsystem must be highly secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Malicious use of the terminal database could generate severe functionality problems in the whole system and money loss. The database must be secured by file system permissions, so the Sys Admin should be the only one allowed to modify the system structure and policies.
The Sys Admin decides if the system allows dynamically assigned terminals or not. In an enterprise network could be desirable not to have dynamic terminals. Thus all the terminals could be assigned statically and the extension numbers would be fixed.

The system shall be designed for making regular maintenance by the sys admin unnecessary.
The robustness is very important because this subsystem is a network server.

R.11.7 Other requirements for Extension-IP translation

The system has the terminal database, which is a core element in the whole system, the database format used for developing is irrelevant, and it could be simple Unix-like ASCII text or any more specialized format.

3.12 System administration: Passwords assignment

Given some services in the telephony system are expensive or desirable to restrict, it is necessary include user rights and user authentication using passwords or codes. The password assignment function manages the establishment of these user rights. The password assignment function shall exist and it must administer and access the user database, a semi-distributed architecture is planned. In this subsection when we express *security restrictions* we mean *allowed and restricted prefixes*.

R.12.1 Inputs required for password assignment

- New group (name, description, prefixes not allowed, prefixes allowed)
- New User (code, name, description, group, terminal, prefixes not allowed, prefixes allowed)
- Edit User (code) or Edit User (name)
- Edit Group (name)
- Delete User (code) or Delete User (name)

R.12.2 Processing required for password assignment

- When system receives *new group* from console, it creates a new empty group. The group has its own security restrictions and automatically all its members have the same security restrictions.
- When receives *New User* from console, the system shall add a new user profile in the users database and register it in proper group. The terminal field could be shared with other users or not, but is highly suggested assign it. That profile lets the system verify permissions for critical parts of the system. The security restrictions applied over users are *added* to the group security restrictions.
- When receives *Edit User* the system opens the profile associated with desired user and allows edition on its properties.
- When receives *Edit Group* the system opens the profile associated with that group and allows edition on its properties: to modify prefixes, members, etc.
- When receives *Delete User* the system deletes the user profile. This operation makes the user unable to use critical resources.

R.12.3 Expected outputs.

- This functional requirement generates confirmation or error messages only. It is through the user interface.

R.12.4 Performance requirements

The edited properties must be applied immediately over critical resources. The application of new properties must not affect calls currently in progress.

R.12.5 Design constraints

The function required must be implemented using ITU H.323 recommendation. This functionality could be added to Gatekeepers for ensuring centralized and easy administration.

R.12.6 Attributes

The subsystem holding this functionality must be highly secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Malicious use of the user database could generate severe functionality problems in the whole system and money loss. The database must be secured by file system permissions, so the Sys Admin should be the only one allowed to modify the user structure and rights.

The subsystem holding this functionality shall be designed for making regular maintenance by the sys admin unnecessary, but must have user interfaces for it.

The robustness is very important because this functionality is in a network server.

R.12.7 Other requirements

The functionality needs a user database, which is a core element in the whole system; the database format used for developing is irrelevant, and it could be simple Unix-like ASCII text or any other specialized format.

The software holding the functionality is suggested to be a network server.

3.13 System Administration: Restricted Prefixes

The critical resources are those scarce or expensive enough for deserving protections, like network bandwidth and links to PSTN. For protecting links to PSTN and other critical resources a *user rights and authentication scheme* is used. Each user calling to any critical resource must provide its code. The code shall be analyzed digit to digit.

R.13.1 Inputs required for restricted frefixes

- Init validation (user)
- Validate number (digit)

R.13.2 Processing.

- When receives *init validation* the system begins a record associated with the user, the system searches for user rights and attributes in *user database* and waits for receiving the digits form user.
- When receives *validate number* the system adds the digit to a preexistent string and matches the same string with the prefix of each element in the list of allowed prefixes by user. If there is any match the system waits for the next digit, else the system matches the string with each element in the list of restricted prefixes. If there is not a list of restricted prefixes associated with the user then the system sends an *access denied* message (the user failed to give an allowed prefix). Else if there is any match then the system sends an *access denied* message and finishes the call (the user pulsed a restricted prefix), else the system waits for the next digit. When the number received is longer than any prefix and is still accepted the system replies the requester with an *Access granted* message.

R.13.3 Outputs.

- Access denied
- Access granted (user characteristics)

R.13.4 Performance requirements

The communications shall use a reliable channel.

R.13.5 Attributes

The subsystem holding this functionality must be highly secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Malicious manipulation of the prefix restriction could result in severe money loss. The program implementing this functionality is recommended to be running in a secure network server.

R.13.6 Other requirements

This subsystem will read the user database. The database format used for developing is irrelevant, and it could be simple Unix-like ASCII text or any other specialized format.

The software holding the functionality is a network server.

3.14 System Administration: Terminal Blocking

In certain circumstances could be necessary to lock *terminal* access to the network. The terminal blocking function lets the sys administrator and terminal owners to disable receiving and making calls.

R.14.1 Inputs required for terminal blocking

- Lock Command (terminal, mode)
- Lock Command (mode)

R.14.2 Processing.

- The system administrator must be able to lock any terminal using the proper interface in Extensions-to-Network-Addresses module. When the system receives *Lock Command (terminal, mode)* from the sys administrator, the indicated terminal is locked using the criteria indicated by *mode.* The criteria are *enabled, incoming call locked, making call locked.* The terminal will be longer blocked until receiving another *Lock Command (terminal, mode)* message with *mode* equal to *enabled.*
- When any terminal receives the command *Lock Command (mode)* from the user, the system sets terminal internal flags in the right state. The main difference is, while the sys admin locking is complete and irrevocable by the user, the terminal locked by user is fully reversible. The terminal may receive the locking command from phone keyboard or from the GUI. The user could undo the locking using the proper command. Another difference is the incoming call feature is not blocked for every source, some *terminals* and *users* are allowed to make calls receivable by *all* terminals.

R.14.3 Expected outputs

- Locked state indication (tone or GUI message)

R.14.4 Performance requirements

The state changes apply immediately in local locking. For sys admin locking the changes apply for any new attempt for establishing or receiving a call.

R.14.5 Design constraints

The function required must be implemented using ITU H.323 recommendation. This functionality shall be added to Gatekeepers and terminals for ensuring centralized and easy administration.

This functionality interacts with many kinds of H.323 terminals, thus it must be able to receive call signaling from all of them. There are GUI commands for software terminals and tones for hardware terminals.

R.14.6 Attributes

The subsystem holding the sys admin related functionality must be highly secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Negligent use of the terminal locking could generate severe functionality problems in the whole system and money loss. The functionality given to the user does not need security.

The sys admin must be able to unlock terminals remotely for preventing long terminal hang-ups.

R.14.7 Other requirements for Terminal blocking

This functionality affects the terminal database, if the message comes from sys admin.

Service implemented by PHNTS.

3.15 System Administration: Group management

The system shall be capable of managing groups. The group is an abstract set of people. The main reason for having groups is to share security restrictions and properties. In **Password assignment** section are explained basic operations for group management. This section speaks about the same groups but related to *phone number aliases*. The system must be capable of assigning a single phone number to an entire group. This means that any member could receive any phone call directed to the group number. Of course, there are policies or priorities for call routing.

R.15.1 Inputs required for Group Management

- Assign group number (group, number)
- Edit priorities (group)

R.15.2 Processing for Group management

- When the system receives *Assign group number* the target group shall receive its shared number. By default, the routing policy will be to send the incoming call to the first registered member. The system routes the call to the terminal owned by the target user, if it has not its own terminal, searches for the next member holding a terminal.
- When receives *Edit priorities*, the system shows the group members and lets the sys admin edit the routing policy for incoming calls. The sys admin establishes certain desired order among members and save it.

R.15.3 Expected outputs

This functional requirement generates confirmation or error messages only. It is through the user interface.

R.15.4 Attributes

The subsystem holding the sys admin related functionality must be highly secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Negligent use of the group management could generate severe functionality problems in the whole system and money loss. The database must be secured by file system permissions, so the Sys Admin should be the only one allowed to modify the system structure and policies.

R.15.5 Other requirements

The subsystem uses and modifies the user database.

3.16 System administration: Dynamic Terminal Management

Normal Terminals (static) have an assigned extension number in the PHNTS service, these terminals are assumed to be always turned on and ready to receive or establish calls. But not all the terminals have these characteristics. Many users could use eventual terminals, like non-resident telephony programs or terminals in portable PCs. The system must be able to assign phone numbers to this kind of terminals. There are two main assumptions: 1) the extension number can not be the same among sessions and 2) the phone number releasing has not warranty to be always notified by dynamic terminals.

R.16.1 Inputs required for dynamic terminal management

- Incoming *Ask for PHNTS* message, from any dynamic terminal.
- Add my terminal to system (network address, owner, description) from any dynamic terminal
- Release terminal by phone or by address, from the network.

R.16.2 Processing.

- When receives any incoming *Ask for PHNTS message from any terminal*: the system replies the terminal with a message indicating its readiness for translating numbers. This message is received from dynamically assigned terminals. These kind of terminal uses that message for detecting PHNTS servers located in the network.
- When receives any *Add my terminal to system*: the system shall be capable of registering a new terminal in the system. The terminal added is assumed to be fully functional and ready to receive or to establish calls. In this process some extra data may be added: terminal description and terminal related permissions as calling and receiving. This input comes from dynamically assigned terminals via network.
- When receives *release terminal* by phone or network address the system deletes the field associated with that terminal in the database. This message only applies over dynamically assigned terminals, deleting a terminal register in the database makes the terminal unable to receive incoming calls from other terminals using alias address (phone number).

R.16.3 Expected outputs

- Message *Ready for making phone translations* for any asking dynamically assigned terminal.
- Added Terminal Confirmation (Extension)

R.16.4 Performance requirements for dynamic terminal management

Changes must be immediately available to any terminal establishing a new phone call. The changes don't affect currently established calls. The system has not other performance requirements. The robustness is very important because this subsystem is a network server.

R.16.5 Design constraints

The subsystem must be H.323 compliant, so the communication shall be implemented using the protocols specified in the ITU recommendation and the final implementation must include all the basic services suggested. Any extra service not specified by ITU may be included in the system, because H.323 specification allows inclusion of non-standard services. The subsystem shall be represented as a H.323 Gatekeeper with added functionality.

The subsystem has not hardware limitations, because it does not interact with any specific hardware beyond the necessary for being executed in its OS and for interacting with TCP/IP protocols stack.

R.16.6 Attributes

The subsystem must be secured. It is recommended to implement and to run the system using a security capable OS like UNIX or Windows NT. Malicious use of the terminal database could generate severe functionality problems in the whole system and money loss. The database must be secured by file system permissions, so the Sys Admin should be the only one allowed modifying the system structure and policies.
The Sys Admin decides if the system allows dynamically assigned terminals or not. In an enterprise network could be desirable not to have dynamic terminals. Thus all the terminals could be assigned statically and the extension numbers would be fixed.

The system shall be designed for making regular maintenance by the sys admin unnecessary.

R.16.7 Other requirements

The system has the terminal database, which is a core element in the whole system, the database format used for developing is irrelevant, and it could be simple Unix-like ASCII text or any more specialized format.

The system shall be capable of detecting terminal absence, if the terminal is dynamically assigned the system will release the associated extension number and will delete the field in terminal database holt by that terminal.

3.17 Billing: Report management

Billing functionality records all the calls done through the system. It is supposed interconnectivity among the system and PSTN exists, thus all the calls must be recorded for being paid later. The calls are registered with user name, group, source terminal, phone number called, starting date/hour, finishing date/hour and origin terminal.

R.17.1 Inputs required for billing report management

- Report (category) from console
- Print (report) from console
- Save (report) from console
- Initialize () from console
- Register call (user, calling terminal, called phone number, time and date of beginning, time and date of finalization) from the network

R.17.2 Processing for billing report management

- When the system receives *Report*, it processes the billing database for getting the desired report. Report categories are, but not restricted to, calls by user, calls by group, calls by terminal. Temporal criteria for searching are allowed too.
- When gets *Print* message, the system prints the report. The system must be able to save reports using permanent storage. The *Initialize* message clears the billing database.
- When receives *Register call*, the system stores the data received in billing database. Phone call receivers like PSTN Gateways and terminals generate this message when the call finishes.

R.17.3 Expected outputs

- Report
- Printed document

R.17.4 Design constraints

The system must implement the database in a format readable by web server CGI-scripts or Java JDBC. This occurs because this functionality is related to *Web vision*.

R.17.5 Attributes

This subsystem must be highly secured. It is recommended to implement and to run the system using a secure OS like UNIX or Windows NT. Negligent use of the billing management could generate severe money loss. The database must be secured by file system permissions, so the Sys Admin should be the only one being allowed to modify accounts.

Given the billing database is very fast growing and important, it requires regular maintenance and it is necessary perform backups and cleanings periodically. Sys Administrator must establish that periodicity.

R.17.6 Other requirements

This subsystem holds the billing database. The billing database is a very important part of System, because it records every phone call.

It is highly suggested run this subsystem in a secure system out of reach of common users. Malicious users making expensive calls could manipulate or reset the billing subsystem.

3.18 Billing: WEB vision

The WWW is a very important media nowadays. The system must have its databases represented in a convenient manner for being used by an HTTP server, that format must allow data retrieving. This subsection describes the billing services provide via Web.

R.18.1 Inputs required for billing web vision

- Report (category)

R.18.2 Processing for billing web vision

- When the system receives *Report*, it watches the billing database for getting the desired report. Report categories are, but not restricted to, calls by user, calls by group, calls by terminal. Temporal criteria for searching are allowed too. The report is returned as standard HTML file.

R.18.3 Expected outputs for billing web vision

- Report as HTML file.

R.18.4 Performance requirements

The processes involved with transactions are required to be "lightweight" as any other WWW server-side process.

R.18.5 Design constraints

The system must be CGI compliant, the platform used for CGI is irrelevant. NCSA HTTP compliant servers or Microsoft IIS servers are recommended. The documents generated as results must be HTML 1.1 compliant, platform-specific operations in HTML must be avoided.

R.18.6 Attributes

This subsystem must be highly secured. It is recommended to implement and to run the system using a secure OS like UNIX or Windows NT. Negligent use of the billing management could generate severe money loss.

Any structural change in the billing database requires changes in the CGI scripts.

R.18.7 Other requirements

This subsystem reads the billing database.

The site running this subsystem must include a fully functional HTTP server. The server must be CGI capable.

Bibliografía.

[1] CHWAN-HWA WU, *Senior Member IEEE*; J. DAVID IRWIN, *Fellow, IEEE*; «Multimedia And Multimedia Communication: A Tutorial». IEEE Transactions on Industrial Electronics. Vol.45 No. 1 February 1998.

[2] DAVE LINDBERG; *PictureTel Corporation*; «The H.324 Multimedia Communication Standard». IEEE Communications Magazine. December 1996.

[3] GARY A. THOM; *Delta Information Systems, Inc.*«H.323: The Multimedia Communications Standard For Local Area Networks». IEEE Communications Magazine. December 1996.

[4] THOMAS J. KOSTAS, MICHAEL S. BORELLA, IKHLAQ SIDHU, GUIDO M. SHUSTER, JACEK GRABIEC, JERRY MAHLER; *3COM* «Real-Time Voice Over Packet-Switched Networks». IEEE Network. January-February 1996.

[5] JOHN F. GIBBON, THOMAS D. C. LITTLE «The Use Of Network Delay Estimation For Multimedia Data Retrieval». IEEE Journal On Selected Areas In Communications. Vol. 14. No. 7. September 1996.

[6] GEROLD BLAKOWASKI, RALF STEINMETZ; *Senior Members, IEEE* «A Media Synchronization Survey:Reference Model, Specification, And Case Studies» ». IEEE Journal On Selected Areas In Communications. Vol. 14. No. 1. September 1996.

[7] JOSEPH Y. HUI, *Senior Member, IEEE*; EZHAN KARASAN, *Student Member, IEEE*; JUN LI, *Student Member*, IEEE; JUNBIAO ZHANG «Client-Server Synchronization And Buffering For Variable Rate Multimedia Retievals» IEEE Journal On Selected Areas In Communications. Vol. 14. No. 1. January 1996.

[8] RALF STEINMETZ «Synchronization Properties In Multimedia Systems» IEEE Journal On Selected Areas In Communications. Vol. 8. No. 3. April 1990.

[9] TEI-WEI KUO, *Member, IEEE*; ALOYSIUS K. MOK. *Member, IEEE* «Incremental Reconfiguration And Load Adjustment In Adaptative Real-Time Systems» IEEE Transactions On Computers. Vol. 46. No. 12. December 1997.

[10] BORKO FURHT, *Florida Atlantic University* «Multimedia Systems: An Overview» IEEE Multimedia. Spring 1994.

[11] THOMAS D. C. LITTLE, *Student Member, IEEE*; ARIF GHAFOOR, *Senior Member, IEEE* «Synchronization And Storage Models For Multimedia Objects» IEEE Journal On Selected Areas In Communications. Vol. 8. No. 3. April 1990.

[12] http://www.comsoc.org.mx/std_voip.html

[13] RALF STEINMETZ; KLARA NAHRTEDT «Multimedia: Computing, Communications & Applications» Prentice-Hall Inc. 1995. ISBN 0-13-324435-0

[14] FÉLIX RAMOS, DAVID GARDUÑO, IVÁN ROMERO, JOSÉ LUIS MÁRQUEZ, « Arquitectura de una Oficina Virtual Distribuida», Presentado en el congreso CIE'98, CINVESTAV México, 1998

[15] DR GEORGE WEIR, *Dept. of Computer Science Livingstone* «Approaches to Multimedia» http://maxwell-11.cs.strath.ac.uk:8001/Multimedia/lect1/index.htm

[16] ANDREW S. TANENBAUM «Redes de computadoras » Prentice Hall • 1999

[17] HENNING SCHULZRINNE «Converging on Internet Telephony» IEEE Internet Computing, Mayo • Junio 1999. Páginas 40-43

[18] DANIELE RIZZETTO, CLAUDIO CATANIA «A voice Over IP Architecture for Integrated Communications» » IEEE Internet Computing, Mayo - Junio 1999. Páginas 53-62

[19] PAWAN GOYAL, ALBERT GREENBERG, CHARLES R, KALMANEK, WILLIAM T. MARSHALL, PARTHO MISHRA, DOUG NORTZ, K. K. RAMAKRISHNAN «Integration of Call Signalin and Resource Management for IP Telephony» » IEEE Internet Computing, Mayo • Junio 1999. Páginas 44-52

[20] CHUNG-MING HUANG, CHIAN WANG «Synchronization for Interactive Multimedia Presentations» IEEE Multimedia, Junio de 1998

[21] DATABEAM CORP. «A primer on the H.323 Series Standard» http://www.databeam.com/h323/h323primer.html

[22] SIEMENS TELECOM NETWORKS «Internet Telephony Tutorial» http://www.webproforum.com/int_tele/index.html

[23] TELOGY NETWORKS «Voice Over Packet Tutorial» http://www.webproforum.com/voice_packet/index.html

[24] BRETT A. LEIDA, M.C. in Electrical Engineering and Computer Science, MIT, USA «A Cost Model of Internet Service Providers: Implications for Internet Telephony and Yield Management» Tesis del MIT, USA - 1999

[25] L. LOGRIPPO, M. FACI, M. HAJ-HUSSEIN «An Introduction to LOTOS: Learning by Examples» University of Ottawa, Protocols Research Group, Department of Computer Science, Ottawa, Ontario, Canada, 1994

[26] ARTURO AZCORRA SALOÑA; JUAN QUEMADA VIVES; SANTIAGO PAVÓN GÓMEZ «An Introduction to LOTOS» Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España., 1995

[27] ARTURO AZCORRA SALOÑA; JUAN QUEMADA VIVES; SANTIAGO PAVÓN GÓMEZ «Design with LOTOS» Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España, 1995

[28] JUAN QUEMADA VIVES; SANTIAGO PAVÓN GÓMEZ, DAVID LARRABEITI LÓPEZ «LOLA: Quick Reference, Version 3R6» Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España, 1995

[29] SANTIAGO PAVÓN GÓMEZ, DAVID LARRABEITI LÓPEZ; G. RABAY «LOLA: User Manual, Version 3R6» Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España. 6 February, 1995

[30] JOSÉ A. MAÑAS; TOMÁS DE MIGUEL; TOMÁS ROBLES; JOAQUÍN SALVACHUA; GABRIEL HUECAS; MARCELINO VEIGA «TOPO: Quick Reference, Front-End, Version

3R6» Departamento de Ingeniería Telmática, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, España. 18 January, 1995

[31] JOSÉ A. MAÑAS; TOMÁS DE MIGUEL; TOMÁS ROBLES; JOAQUÍN SALVACHUA; GABRIEL HUECAS; MARCELINO VEIGA «TOPO: Quick Reference, Auxiliary Tools, Version 3R6» Departamento de Ingeniería Telmática, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, España. 17 October, 1994

[32] JOSÉ A. MAÑAS; TOMÁS DE MIGUEL; TOMÁS ROBLES; JOAQUÍN SALVACHUA; GABRIEL HUECAS; MARCELINO VEIGA «TOPO: Quick Reference, C Code Generator, Version 3R6» Departamento de Ingeniería Telmática, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, España. 20 January, 1995

[33] JOSÉ A. MAÑAS; MARCELINO VEIGA «How to Use LOTOS Data Types from C Code and How to Implement them by Hand Using TOPO, Version 3R6» Departamento de Ingeniería Telmática, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, España. 10 October, 1994

[34] PETER VAN EIJK; HARRO KREMER; MARTEN VAN SINDEREN «On the use of Specification styles for automated protocol implementation from LOTOS to C» University of Twente, Fac. Informatics, AE Euschede, NL, June 19, 1991

[35] L. LOGRIPPO; MAZZEN HAJ-HUSSEIN «Specifying Distributed Algorithms in LOTOS» University of Ottawa, Protocols Research Group, Department of Computer Science, Ottawa, Ontario, Canada, 1994

[36] L. LOGRIPPO; M. FACI; M. HAJ-HUSSEIN «Specifying Features and Analysing their Interactions in a LOTOS Enviroment» University of Ottawa, Protocols Research Group, Department of Computer Science, Ottawa, Ontario, Canada, 1994

[37] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.323, Packet-Based Multimedia Communications Systems» 1996

[38] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.225.0, Media Stream Packetization and Synchronization for Visual Telephone Systems on non-guaranteed Quality of Service LAN's» 1998

[39] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.245, Control Protocol for multimedia communications» 1998

[40] CCITT «Rec. G.711, Pulse Code Modulation (PCM) of voice frequencies» 1988

[41] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. G.723.1, Speech Coders: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbps» 1996

[42] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.261, Video Codec for Audiovisual Services at p X 64 kbps» 1993

[43] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. T.120, Transmission Protocols for Multimedia Data» 1996

[44] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.320, Narrow-Band visual Telephone Systems and Terminal Equipment» 1996

[45] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. H.324, Terminal for Low Bit Rate multimedia communications» 1996

[46] INTERNATIONAL TELECOMMUNICATIONS UNION – TELECOMMUNICATIONS «ITU-T Rec. Q.931, ISDN User-Network Interface Layer 3 Specification for Basic Call Services» 1996

[47] FÉLIX RAMOS, DAVID GARDUÑO, IVÁN ROMERO «The 1999 International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'99» Proceedings of the International Conference on Parallel and distributed Processing Techniques and Applications, PDPTA'99, pp. 60-66, ISBN 1-892512-09-2. Las Vegas, Nevada, USA; July 1999

[48] Romero Hernández Ivan «Algoritmos de sincronización de presentaciones multimedia distribuidas» Tesis de M. C. A presentarse en 2000 en el CINVESTAV-GDL

El Jurado designado por el Departamento de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "UNA ARQUITECTURA DE SOFTWARE CONSIDERANDO MECANISMOS DE SINCRONIZACIÓN PARA SISTEMAS MULTIMEDIA BASADOS EN LA RECOMENDACIÓN ITU – T H.323" el día 10 de Marzo de 2000.

Dr. José Luis Leyva Montiel
Investigador CINVESTAV 3B
CINVESTAV DEL IPN
Guadalajara

Dr. Deni Torres Roman
Investigador CINVESTAV 2C
CINVESTAV DEL IPN
Guadalajara

Dr. Ricardo Raúl Jacinto Montes
Investigador CINVESTAV 2B
CINVESTAV DEL IPN
Guadalajara