Capítulo 1: Unidad multipunto software para videoconferencia H.323.

1.1 Introducción

El objetivo de este proyecto es gestionar la actividad de una unidad multipunto software para videoconferencia H.323 que controle la comunicación mediante el programa *NetMeeting* entre varios usuarios, para lo que se modificará el código de libre distribución de una unidad de control multipunto basada en H.323 hasta adaptarlo a las necesidades del proyecto.

Se puede definir una videoconferencia como un sistema de comunicación diseñado para llevar a cabo encuentros a distancia, el cual nos permite la interacción visual, auditiva y verbal con personas de cualquier parte del mundo, siempre y cuando los sitios a distancia tengan equipos compatibles y un enlace de transmisión entre ellos.

Una Unidad de Control Multipunto (MCU, *Multipoint Control Unit*) es la unidad funcional de una infraestructura de videoconferencia que permite que varias personas se comuniquen simultáneamente en una habitación virtual de multiconferencia, pudiendo gestionar a la vez distintas habitaciones.

Una MCU H.323 crea una videoconferencia punto a punto con cada terminal que intervenga en la comunicación. Combina las entradas en un entorno compartido como si fuera un espacio físico de encuentro, que se conoce como habitación virtual.

El proyecto tendrá como base la recomendación H.323, es un estándar de la Unión Internacional para las Telecomunicaciones (ITU, *International Telecommunications Union*) que proporciona información para comunicaciones multimedia sobre redes LAN o Internet en tiempo real. Los usuarios pueden conectarse con otras personas por Internet y usar varios productos que soporten H.323. El estándar H.323 define cómo se formatea la información de audio y video y cómo se empaqueta para su transmisión por la red.

En el proyecto se trabajará principalmente con dos programas: *NetMeeting* y *OpenMCU*, aunque a la hora de realizar pruebas también se han usado otros programas

clientes, desarrollados por la organización *OpenH323 Project*, como son *OhPhone* para Linux y *OpenPhone* para Windows.

Básicamente cada ordenador que desee establecer o entrar en una multiconferencia hará una llamada mediante el programa *NetMeeting* al ordenador en el que se está ejecutando el programa *OpenMCU* modificado, el cual se encargará de unirlo a una multiconferencia concreta o crear una nueva dependiendo del llamante. Todos los participantes de una misma multiconferencia estarán en la misma habitación virtual, pudiendo haber varias habitaciones creadas al mismo tiempo.

Se eligió *NetMeeting* ya que proporciona una solución para videoconferencias en Internet a los usuarios de Windows, y Microsoft desarrolló sus características basándose en la infraestructura definida en el estándar H.323, lo que permite que *NetMeeting* interactúe con otros productos basados en H.323.

Para el desarrollo del proyecto se parte del código proporcionado por la organización *OpenH323 Project*, la cual pretende crear una implementación operativa del protocolo de videoconferencia H.323 con código fuente libre para su uso y modificación. *OpenMCU* es la unidad de multiconferencia software que usa el protocolo H.323 de esta organización, trabaja estableciendo un proceso escucha en un ordenador y luego esperando las llamadas entrantes de los ordenadores que pretenden establecer o crear una multiconferencia. Cuando una conexión entrante se establece, se determina a qué videoconferencia (habitación virtual) de las ya establecidas se une para añadirlo a la misma, o si hay que crear una nueva habitación para una nueva multiconferencia.

El código de *OpenMCU* está escrito en C++, por lo que se seguirá la metodología orientada a objetos. Partiendo de este código se añadirán nuevas funcionalidades al programa para adaptarlo a las necesidades de este proyecto, como se explicará en el capítulo 4 detalladamente.

Para probar el sistema se han necesitado como mínimo tres ordenadores en red, aunque para tener al menos dos multiconferencias funcionando a la vez se han necesitado cinco. Uno de los ordenadores tendrá la MCU y el resto serán los usuarios que establezcan las multiconferencias, éstos deberán estar provistos del hardware (tarjeta de sonido, micrófono, altavoces, webcam) y del software (*NetMeeting*, *OhPhone*, u *OpenPhone*) necesarios para la comunicación.

Este trabajo formará parte del proyecto CIMA (Centro de Información Multimedia Avanzado), desarrollado por el Departamento de Tecnología Electrónica, E.T.S.I. Telecomunicación, el cual consiste básicamente en un sistema proveedor de servicios de teleasistencia sobre IP. El elemento central de este proyecto es el distribuidor de llamadas automático (ACD, *Automatic Call Distributor*), que se encarga de gestionar y distribuir las llamadas que realizan unos clientes que intentar comunicarse con unos asesores que les dan servicio mediante videoconferencia. Las peticiones de los clientes se envían a través de una página web.

Entre las líneas de desarrollo a seguir de este proyecto estaba prevista la inclusión de una MCU H.323 para permitir el establecimiento de sesiones multipunto. La MCU permite establecer multiconferencias entre clientes y asesores.

Esquemáticamente la arquitectura del sistema se representa en la siguiente figura:

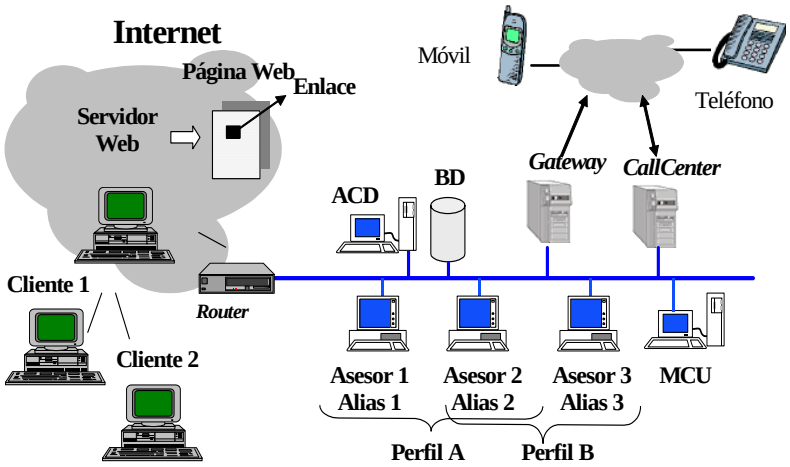

Figura 1-1. Arquitectura del sistema

De forma general los clientes harán una petición HTTP al ACD, el cual decidirá qué asesor atiende a cada cliente en función de la información que tendrá almacenada en una base de datos. Usuario y asesor harán una llamada H.323 a la OpenMCU modificada, que les asignará la habitación en la que previamente el ACD ha decidido que se realice la sesión, estableciendo una sala de multiconferencia para ambos. Si un segundo cliente realiza otra petición de asistencia, el ACD puede decidir que se una a la sala previamente creada para el asesor y el primer cliente, con lo que lo unirá a la multiconferencia, o puede decidir que lo atienda otro asesor en otra multiconferencia, o bien denegarle el acceso. La MCU actuará de una forma u otra dependiendo del fichero de acceso que cree el ACD para cada cliente.

1.2 Descripción de los contenidos de cada capítulo:

La documentación del proyecto se dividirá en los siguientes capítulos:

- En el capítulo 2 se hace una introducción a la recomendación H.323 centrándose básicamente en los elementos que integran un sistema H.323 y los protocolos que esta recomendación usa.

- En el capítulo 3 se explica el entorno en el que se trabaja, dando una introducción al proyecto CIMA, y también se explica el proyecto *OpenH323* y se hace una breve introducción a las librerías *PWLib* y *OpenH323* necesarias para desarrollar una implementación del protocolo H.323.

- El capítulo 4 se centra en la MCU y las modificaciones realizadas al código desarrollado por *OpenH323*, para adaptarlo a las necesidades del proyecto. En este capítulo se explica también la interfaz gráfica y las librerías QT utilizadas para su implementación. Además se describen las pruebas realizadas.

- El capítulo 5 recoge las conclusiones.

Por último se incluyen los apéndices y la bibliografía consultada:

- Apéndice A: se centra en la arquitectura de *OpenH323*.

- Apéndice B: da una visión de la arquitectura de la MCU.

- En el apéndice C se describen los recursos utilizados para el desarrollo y pruebas del proyecto.

- En el apéndice D se indican los pasos para la instalación.

- El apéndice E consiste en el manual de usuario.

- Apéndice F: lista de acrónimos y definiciones.

- Apéndice G: muestra la bibliografía consultada.

Capítulo 2: H.323

2.1 Introducción

La ITU-T fue el primer comité de estandarización que desarrolló un estándar para la transmisión de tráfico multimedia sobre redes de paquetes. El estándar denominado H.323 se publicó en 1996 con el nombre "Sistemas y terminales de telefonía visual sobre redes de área local sin garantías de calidad de servicio" (*Visual Telephone Systems and Equipment for LANs which provide a non-guaranteed Quality of Service*). La principal aportación de este estándar fue el desarrollo de un conjunto de protocolos de señalización que permiten controlar el establecimiento, mantenimiento y liberación de conexiones multimedia (audio, vídeo y datos) sobre redes de paquetes, ya que los protocolos para la transmisión de estos medios fueron adoptados de trabajos previos, principalmente desarrollados por el IETF (Internet Engineering Task Force), como los protocolos RTP y RTCP. Tras esta primera versión apareció la segunda en 1998, H.323v2, con un nuevo nombre: "Sistemas de comunicación multimedia sobre redes de conmutación de paquetes" (*Packet based multimedia conmmunications system*), nombre que permanece hasta la actualidad. La versión 4 fue aprobada en noviembre de 2000.

La recomendación H.323 de la Unión Internacional de Telecomunicaciones (ITU) se ocupa de la transmisión de servicios de comunicación multimedia sobre redes de paquetes, que pueden no proporcionar una calidad de servicio (QoS).

En la recomendación H.323 se ha producido una gran colaboración entre los fabricantes de ordenadores y de equipos de telecomunicación. De esta forma, la adopción del estándar H.323 asegura un alto grado de funcionalidad e interoperatibilidad de estos equipos. Entre las compañías vinculadas al mundo de los ordenadores que proporcionan su soporte a la recomendación H.323 cabe destacar *Intel*, *Microsoft* y *Netscape*. Ello permite un mayor grado de conocimiento del estándar por parte del mercado. Esta situación de conocimiento, unida a la capacidad de comunicarse entre equipos de diferentes fabricantes conectados a diferentes tipos de redes, ha sido el factor clave del éxito de la recomendación H.323.

El tipo de red basada en paquetes sobre la que es posible establecer una comunicación mediante H.323 puede incluir redes de área local, redes de área metropolitana, intrarredes e interredes (p.e. Internet). H.323 no describe el interfaz de red, la red física ni el protocolo

usado en la red, pero puede funcionar con multitud de ellos. Los tipos de redes más frecuentes que pueden usarse son:

- Ethernet: IEEE 802.3, LAN que trabaja con el método de control de acceso al medio (bus) acceso múltiple por detección de portadora, con detección de colisiones (*Carrier sense, múltiple access with colision detect*,CSMA-CD).

- Fast Ethernet: IEEE 802.3u, LAN de alta velocidad compatible con las redes CSMA/CD.

- FDDI: ISO 9314, Interfaz de datos distribuida por fibra (*Fiber Distributed Data Interface*), red de anillo basada en fibra óptica que puede servir como LAN o MAN de alta velocidad.

- Token Ring: IEEE 802.5, Lan de tipo anillo.

- ATM: Modo de transferencia asíncrono, usado en redes LAN, MAN y WAN.

2.2 Elementos que integran un sistema H.323

Los elementos que integran un sistema H.323 son:

- **Terminal:** Proporcionan comunicaciones en tiempo real de audio y opcionalmente vídeo y/o datos.

- ***Gateway*:** punto final de la red que proporciona comunicaciones bidireccionales en tiempo real entre terminales H.323 sobre red basada en paquetes y otros terminales ITU en la red de conmutación de circuitos (por ejemplo H.310, H.320, H.321, H.322, H.324, H.324M y V.70) o con otra pasarela (*gateway*) H.323.

- ***Gatekeeper*:** Sistema H.323 en la red que proporciona la traducción de direcciones y controla los accesos a la red de los terminales H.323, *gateways* y MCU. Puede gestionar el ancho de banda.

- **Controlador Multipunto (MC):** Sistema H.323 en la red que proporciona el control para tres o más terminales involucrados en una conferencia multipunto. Negocia las capacidades entre los diferentes terminales pero no puede mezclar o conmutar audio, vídeo o datos.

- **Procesador Multipunto (MP):** Sistema H.323 en la red que centraliza el procesador de audio, vídeo y/o datos en conferencias multipunto. Realiza las mezclas y conmutaciones de éstos bajo el control del MC.

- **Unidad de Control Multipunto (MCU):** Punto final en la red que proporciona la capacidad para tres o más terminales y *gateways* para participar en una conferencia multipunto. Contiene un MC y opcionalmente un MP.

Los terminales, MCUs y pasarelas se denominan *endpoint* H.323. Los *endpoint* pueden ser llamados.

2.2.1 Terminales H.323

Es un terminal de la red que proporciona en tiempo real comunicación bidireccional con otro terminal H.323, pasarela o MCU (Unidad de Control Multipunto). Un terminal debe soportar al menos transmisión de voz, voz y datos, voz y vídeo, o voz, datos y vídeo.

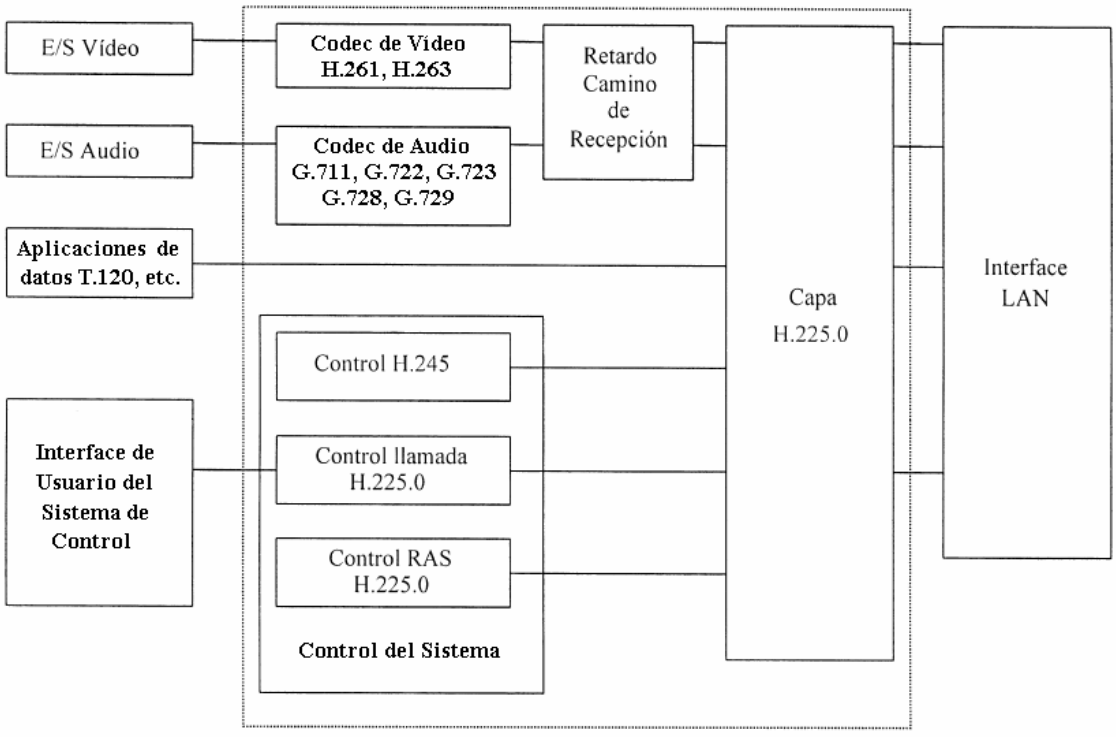

Figura 2-1. Terminal H.323

Un ejemplo de terminal H.323 se muestra en la figura 2-1, todos los terminales deben tener una unidad de control del sistema, una capa H.225.0, interfaz de red y una unidad de

codec de audio, la unidad de codec de vídeo y la de aplicaciones de datos de usuario son opcionales.

2.2.1.1 Interfaz de la red basada en paquetes

Es de implementación específica y está fuera del ámbito de la recomendación H.323. Sin embargo, debe proporcionar los servicios descritos en la recomendación H.225.0, lo cual incluye lo siguiente:

- Servicio seguro (por ejemplo TCP o protocolo de control de transporte, SPX o intercambio de protocolo secuencial) entre extremos de la comunicación para el control del canal H.245, los canales de datos y el canal de señalización de llamadas.

- Servicio inseguro (por ejemplo UDP o protocolo de datagrama de usuario, IPX o intercambio de protocolo de interred) entre extremos de la comunicación para los canales de audio, de vídeo y de RAS (Registro, admisión y estatus).

2.2.1.2 Unidad de codificación de vídeo

Es un elemento opcional, en caso de incorporarse permite realizar la codificación del equipo de vídeo y decodifica la señal de vídeo presente a la entrada, proporcionando una salida hacia el visualizador de vídeo.

Si el terminal H.323 proporciona comunicaciones de vídeo, debe soportar como mínimo H.261 QCIF, siendo opcionales los otros modos de H.261 o H.263.

Estándar	**Año**	**Características**
H.261	1990	Estándar de codificación de vídeo. Soporta formatos de imagen QCIF (176x144 pixeles) y opcionalmente CIF (352x288 pixeles)
H.263	1996	Estándar de codificación de vídeo. Soporta formatos de imagen SQCIF (128x96 pixeles), QCIF y opcionalmente: CIF, 4CIF (70x576 pixeles), 16CIF (140x1152 pixeles)

Tabla 2.1. Codecs de vídeo

La negociación con el terminal H.323 receptor mediante H.245 permite transmitir otros estándares de canal de vídeo.

La transferencia de vídeo, el formato de la imagen y las opciones del algoritmo de codificación que puede aceptar el receptor se definen durante el intercambio de funciones mediante H.245, de forma que el codificador tiene libertad para transmitir en cualquier condición que sea interpretable por el receptor. Además, el receptor puede enviar peticiones (mediante H.245) para un determinado modo, sin embargo el receptor puede ignorar estas peticiones. Además, dos terminales H.323 pueden establecer comunicaciones con transferencias asimétricas, diferentes resoluciones y número de imágenes por segundo. Un ejemplo sería un terminal H.323 enviando imágenes CIF y recibiendo QCIF, pudiendo ser además con una codificación de vídeo diferente, así como la transmisión y recepción simultánea de un número de imágenes por segundo distinto.

El modo de trabajo seleccionado se indica al receptor en el mensaje OpenLogicalChannel de H.245, mientras que el formato del flujo de bits de vídeo queda definido por la recomendación H.225.0.

2.2.1.3 Unidad de codificación de audio

Se encarga de codificar la señal de audio presente en la entrada de forma que sea posible una transmisión más eficiente, y decodifica la señal de audio que recibe del otro terminal con el que ha establecido la comunicación.

Todos los terminales H.323 deben ser capaces de codificar y decodificar voz de acuerdo con la recomendación G.711 según la ley A (estándar europeo) y la ley u (estándar americano). El resto de estándares mostrados en la tabla, así como la codificación de audio de MPEG 1 son opcionales.

Estándar	**Año**	**Velocidad**	**Características**
G.711	1988	64 Kb/s	PCM en el margen de frecuencias de voz.
G.722	1988	64 Kb/s	Codificación de audio (por subbandas) con ancho de banda 7KHz.

G.723.1	1996	5.3 y 6.3 Kb/s	CELP algebraico y multipulso.
G.728	1992	16 Kb/s	LD-CELP (CELP de bajo retardo).
G.729	1996	8 Kb/s	CS-ACELP (CELP algebraico de estructura conjugada).

Tabla 2.2. Codecs de audio

El algoritmo de codificación se fija en el intercambio de funciones entre equipos emisor y receptor mediante la recomendación H.245. Al igual que la transmisión de vídeo, la comunicación puede ser asimétrica (por ejemplo enviar audio codificado con G.711 y recibir en G.729, siempre y cuando ambos equipos lo soporten).

El formato de la señal de audio viene estipulado por la recomendación H.225.0, y es posible enviar y/o recibir más de un canal de audio simultáneamente, lo cual es interesante, por ejemplo, para realizar transmisiones en dos idiomas distintos simultáneamente.

Cuando se trabaja a bajas velocidades de transferencia (< 56 Kb/s) no es posible usar la codificación de audio G.711. En dicho caso debe usarse la recomendación G.723.1 o la G.729.

2.2.1.4 Retardo del camino de recepción

Este bloque se encarga de mantener la sincronización y tener en cuenta el retardo que experimentan los paquetes en su transmisión por la red. Otro aspecto importante es mantener la sincronización entre las imágenes y el sonido, de forma que si en la imagen aparece una persona hablando, el movimiento de los labios corresponda con los sonidos reproducidos por el altavoz.

2.2.1.5 Aplicaciones de datos de usuario

Es un elemento opcional que soporta aplicaciones telemáticas como la transferencia de imágenes estáticas, intercambio de ficheros, acceso a bases de datos, etc. Puede existir uno o más canales de datos, los cuales pueden ser unidireccionales o bidireccionales, dependiendo del tipo de aplicación. Por defecto se usa la recomendación T.120, pero son posibles otras. Por ejemplo, es posible controlar la cámara del sistema remoto (zoom, panorámico, encuadre, etc.) a partir de los protocolos H.281 y H.224.

2.2.1.6 Unidad de control del sistema

Proporciona la señalización para que el terminal H.323 funcione de forma adecuada. Realiza el control de llamadas, el intercambio de funciones entre los equipos implicados en la comunicación, la señalización de los canales lógicos de comandos e indicaciones, y emplea mensajes para abrir y describir completamente el contenido de los canales lógicos.

2.2.1.7 Función de control H.245

La función de control H.245 (protocolo de control para las comunicaciones multimedia) utiliza el canal de control H.245 para llevar mensajes de control sobre el modo de trabajo del equipo H.323. Entre ellos cabe destacar: intercambio de capacidades entre equipos, apertura y cierre de canales lógicos, peticiones de preferencia de modos, mensajes de control de flujo e indicaciones y comandos generales.

2.2.1.8 Función de señalización RAS (Registration, Admisión, Signaling)

Utiliza mensajes de H.225.0 para realizar registros, admisiones, cambios de anchos de banda, estado y liberación entre puntos finales y *gatekeepers*. El canal de señalización RAS es independiente del canal de señalización de llamadas y del canal de control H.245. En entornos de red sin *gatekeeper* no se usa el canal de señalización RAS, mientras que, si existe *gatekeeper*, el canal está abierto entre el *gatekeeper* y el punto final. El canal de señalización RAS es el primero que se abre entre puntos finales H.323.

2.2.1.9 Función de señalización de llamadas

El canal de señalización de llamadas es independiente del canal de señalización RAS. En sistemas sin *gatekeeper* el canal de señalización de llamadas está abierto entre los dos puntos finales implicados en la llamada. Si existe *gatekeeper*, el canal de señalización de llamada está abierto entre el punto final y el *gatekeeper*, o entre puntos finales según la elección de este último.

2.2.1.10 Capa H.225.0

Da formato a los datos, audio, vídeo y flujos de control para enviarlos a la interfaz de red, y restaura los datos, audio, vídeo y flujos de control recibidos de la interfaz de red.

Además realiza el entramado lógico, la numeración de secuencias, y la detección y corrección de errores de forma apropiada a cada tipo de medio.

Los canales de vídeo, audio datos o informaciones de control se establecen de acuerdo con la recomendación H.245 (protocolo de control para comunicaciones multimedia), y son unidireccionales e independientes en cada dirección de transmisión. Algunos canales lógicos, por ejemplo para datos, pueden ser bidireccionales. Se puede transmitir cualquier número de canales de cada tipo de medio, pero para cada llamada existe un único canal de control H.245. Además de los canales lógicos, los puntos finales H.323 usan dos canales de señalización para el control de llamadas y las funciones relacionadas con el *gatekeeper*. El formato de los datos enviados por los canales debe cumplir la recomendación H.225.0.

Los canales lógicos reciben un número de canal lógico en el margen 0 a 65535 que es seleccionado de manera arbitraria por el transmisor, excepto el canal 0 que se asigna de forma permanente al canal de control H.245.

El límite superior de la tasa de bits del canal lógico se establece a partir de las limitaciones de los terminales implicados en la comunicación, de forma que será el menor de ellos.

2.2.1.11 Interoperabilidad con otros terminales de transmisión de datos, audio y/o vídeo

Los terminales H.323 pueden comunicarse con terminales de otras normativas conectados a la misma u otra red, según se muestra en la figura 2-2.

Los principales equipos que pueden establecer una comunicación con un terminal H.323 son:

- V.70: Son equipos para la transmisión simultánea de voz y datos sobre la Red Telefónica Conmutada (RTC).
- De voz: Son los teléfonos convencionales conectados a la RTC o los teléfonos digitales conectados a una RDSI.
- H.310: Terminales y sistemas de comunicaciones audiovisuales conectados a RDSI de banda ancha.

- H.320: Sistemas de videoteléfonos sobre RDSI de banda estrecha.
- H.321: Adaptación de los videoteléfonos a entornos de RDSI de banda ancha.
- H.322: Videoteléfonos para LAN de calidad de servicio garantizada (ancho de banda garantizado).
- H.324: Terminales para comunicaciones multimedia a bajas velocidades.

La mayoría de ellos requieren el uso de un *gateway* que realice la adaptación. Un *gateway* para adaptar a H.323 no tiene por qué implementar todas las conversiones posibles. Por tanto, a la hora de decidirse por incorporar uno habrá que escogerlo de forma que permita la comunicación con todos los otros tipos de terminales que se dispongan.

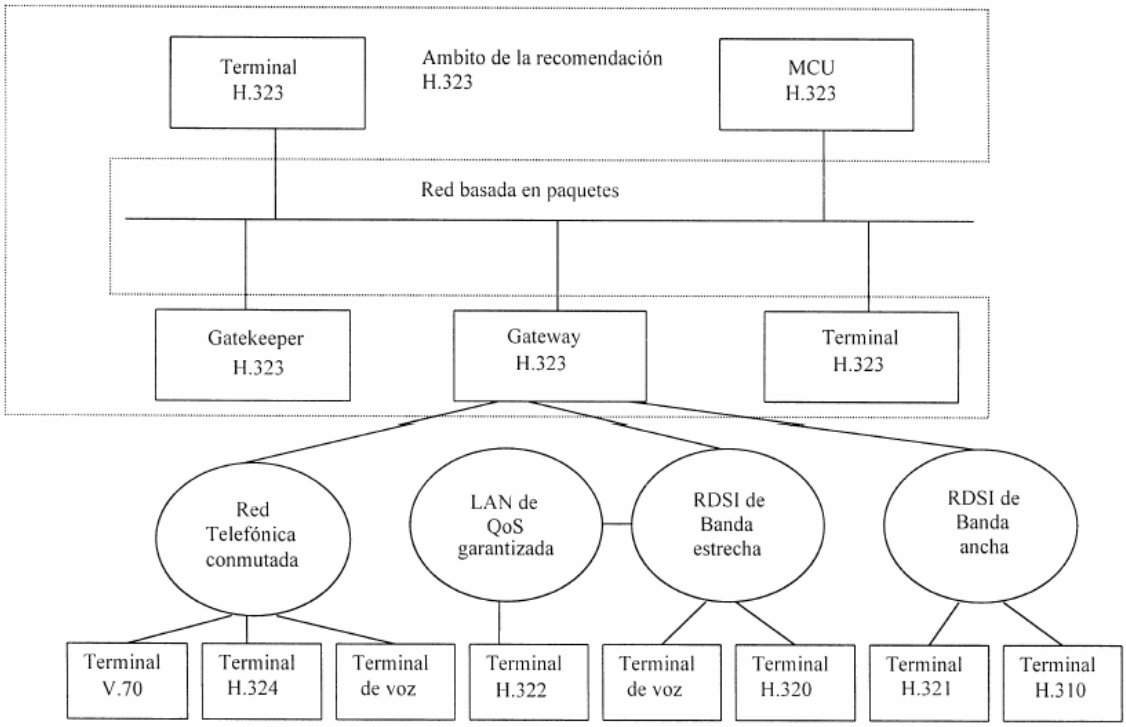

Figura 2-2. Interoperabilidad de otros terminales con terminales H.323

2.2.2 Pasarela H.323 (Gateway)

Es un elemento que permite interoperar a los terminales H.323 con terminales en otros tipos de redes. Las pasarelas se conectan directamente con terminales H.323 o bien con otras

pasarelas o terminales en otras redes, y realizan las funciones de adaptación entre flujos de información, así como entre los protocolos de control en ambos entornos.

La pasarela es un elemento opcional en una conferencia H.323. Proporciona servicios de traducción entre terminales H.323 y otros terminales ITU. Estos servicios incluyen la traducción entre distintos formatos de transmisión (por ejemplo entre H.225.0 y H.221), y entre procedimientos de transmisión (por ejemplo entre H.245 y H.242). Esta traducción se especifica en la recomendación H.246. Además el g*ateway* también traduce entre los codecs de video y audio usados en ambas redes, y procesa la configuración de la llamada y limpieza de ambos lados de la comunicación.

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

En general, el propósito del *gateway* (cuando no opera como una MCU) es reflejar las características de un punto final en la red basada en paquetes en el punto extremo en la Red de Circuitos Conmutados (SCN) y al contrario, de forma transparente.

Una pasarela está compuesta por:

- Control de pasarela de medios (MGC, *Media Gateway Controller*)
- Pasarela de medios (MG, *Media Gateway*)

El MGC y el MG distinguen entre elementos de administración de recursos de alto y bajo nivel. El MGC es responsable de los elementos de alto nivel, como la disponibilidad de recursos tales como canceladores de eco, pero no asigna recursos específicos a sesiones específicas. El MG es responsable de la gestión y reparto de los elementos de bajo nivel, tales como las manipulaciones hardware necesarias para conmutar y procesar las cadenas multimedia en la pasarela de medios.

2.2.3 Gatekeeper

El *gatekeeper* proporciona servicios de control de llamada al resto de terminales H.323. Es un elemento opcional de la arquitectura. Una red puede tener más de un *gatekeeper*, y éstos pueden comunicarse entre sí, sin embargo los protocolos de comunicación utilizados entre dos o más *gatekeeper* no están especificados en la recomendación.

El *gatekeeper* está separado lógicamente de los terminales, pero su implementación física puede coexistir con un terminal, MCU, pasarela, MC o cualquier dispositivo de la red que no pertenezca a la recomendación H.323.

Cuando está presente debe proporcionar los siguientes servicios:

- Traducción de direcciones: Una de las principales funciones es traducir un número telefónico o un alias a la dirección de transporte apropiada, para ello el *gatekeeper* debe disponer de una tabla de traducción de direcciones, la cual se actualiza cada vez que un dispositivo se registra o se borra en el *gatekeeper*.

- Control de admisión: El *gatekeeper* debe autorizar o negar el acceso a la red H.323 utilizando mensajes descritos en la recomendación H.225.0 (ARQ/ACF/ARJ). El control de admisión se puede basar en la autorización de la llamada, ancho de banda o cualquier otro criterio que estimen los fabricantes. También puede ser una función nula que admita todas las peticiones.

- Control de ancho de banda: El *gatekeeper* debe soportar los mensajes H.225.0 respecto a la asignación de ancho de banda (BRQ/BRJ/BCF). Mediante los protocolos adecuados, puede indicar a cada punto extremo el ancho de banda total disponible según el tipo de llamada, las categorías de terminales, etc. También puede ser una función nula que acepte todas las peticiones para los cambios en el ancho de banda.

- Gestión de su zona: Un *gatekeeper* define una zona H.323. Los terminales, pasarelas y MCUs registrados en el mismo *gatekeeper* pertenecen a la misma zona. El *gatekeeper* debe brindar como mínimo los servicios descritos anteriormente para todos los terminales, MCUs y pasarelas de su zona.

De forma adicional a los servicios indicados, el *gatekeeper* puede proporcionar algunos otros opcionales como:

- Señalización para el control de llamadas: el *gatekeeper* puede elegir que la señalización para el establecimiento y liberación de llamadas se realice directamente entre dos terminales o a través de él mismo, de esta forma el *gatekeeper* puede evitar gestionar las señales de control de llamada H.225.0.

- Autorización de llamadas: Mediante el uso de señales H.225.0 el *gatekeeper* puede autorizar o negar llamadas solicitadas desde los terminales. Las razones para autorizar o negar llamadas puede incluir criterios de restricciones de ciertos terminales o pasarelas, restricciones en determinados horarios del día, etc. La recomendación no establece cuales deben ser estos criterios para que cada fabricante los adapte a sus necesidades.

- Gestión del ancho de banda: control del número de terminales H.323 a los que se les permite el acceso simultáneo a la red. Mediante el uso de señales H.225.0 el *gatekeeper* puede decidir no aceptar llamadas debido a limitaciones del ancho de banda. El criterio para establecer cuando hay suficiente ancho de banda no se establece en la recomendación. Puede ser una función nula de forma que todos los terminales tengan concedido el acceso.

- Gestión de llamadas: por ejemplo, el *gatekeeper* puede mantener una lista de las llamadas H.323 en curso. Esta información puede ser necesaria para indicar si un terminal está ocupado, y para proporcionar información a la función que gestione el ancho de banda.

El *gatekeeper* puede elegir recibir los canales de control H.245 de dos terminales en conferencia punto a punto. Cuando la conferencia se convierte en multipunto, el *gatekeeper* puede reexpedir los canales de control H.245 a un MC. El *gatekeeper* no necesita procesar las señales H.245, sólo tiene que pasarlas entre los terminales o entre los terminales y el MC.

Las redes que contengan una pasarela deben también contener un *gatekeeper* para traducir las direcciones entrantes en direcciones de transporte.

Las entidades H.323 que contengan un *gatekeeper* deben tener un mecanismo para desactivarlo, de forma que cuando haya varias entidades H.323 que contengan un *gatekeeper* en la red, estas puedan configurarse dentro de la misma zona.

2.2.4 Controlador Multipunto (MC, Multipoint Controller)

El MC proporciona las funciones de control para sostener conferencias entre 3 o más puntos extremos en una conferencia multipunto. Lleva a cabo el intercambio de capacidades con cada *endpoint* en una conferencia multipunto, éste envía un conjunto de capacidades al

terminal en la conferencia que indica los modos de operación en los que ella puede transmitir. El MC puede revisar el conjunto de capacidades que envía a los *endpoint* como resultado de la incorporación de terminales, del abandono de la conferencia de alguno de éstos, o por otros motivos.

De esta forma, el MC determina el Modo de Comunicación Seleccionado (SCM, *Selected Communications Mode*) para la conferencia. El SCM puede ser común para todos los *endpoint* en la conferencia. En cambio, algunos *endpoint* pueden tener un SCM distinto al de otros en la conferencia. La forma en la que el MC determina un SCM no está dentro del ámbito de esta recomendación.

Como parte de la organización de una conferencia multipunto, un *endpoint* se conectará al MC utilizando su canal de control H.245. Esta conexión puede ocurrir:

- Vía una conexión explícita con una MCU.
- Vía una conexión implícita al MC en un g*atekeeper*.
- Vía una conexión implícita al MC en otro terminal o pasarela en la conferencia multipunto.
- Vía una conexión implícita a través de un *gatekeeper* a una MCU.

La elección de un modo de conferencia (por ejemplo, descentralizada o centralizada) ocurre después de la conexión en el MC usando señalización H.245. La elección del modo de conferencia puede ser limitado por la capacidad del *endpoint* o del MC.

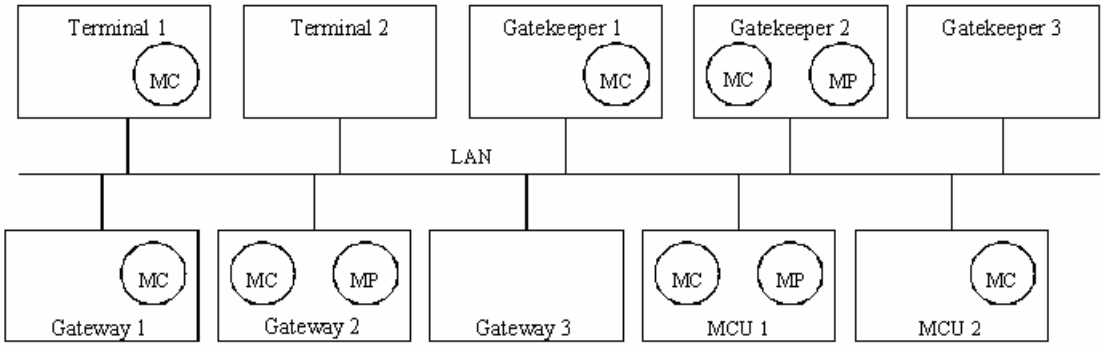

Figura 2-3. Posibles ubicaciones de MC y MP en un sistema H.323

Se puede ubicar el MC en un g*atekeeper*, pasarela, terminal o MCU, tal y como se muestra en la figura 2-3.

Un MC en un terminal no puede ser llamado. Puede ser incluido en la llamada para procesar la señalización H.245 para mantener conferencias multipunto ad hoc. En este caso, puede no haber diferencia entre el MC y la función de control H.245 del terminal. Las comunicaciones entre ellos están fuera del ámbito de esta recomendación.

Un MC en un gatekeeper no puede ser llamado; sin embargo, una MCU en un g*atekeeper* puede ser llamada. Una MCU en un g*atekeeper* puede funcionar como una MCU independiente. Un MC en un *gatekeeper* puede ser usado para mantener conferencias multipunto ad hoc cuando un g*atekeeper* recibe los canales de control H.245 de los terminales. De esta forma, el g*atekeeper* puede encaminar los canales de control H.245 al MC al comienzo de la llamada o cuando la conferencia se vuelva multipunto.

El *gateway* puede funcionar como un terminal o como una MCU. Cuando funciona como terminal puede contener un MC. En este caso tiene las mismas características que se han descrito arriba para un MC en un terminal.

Una MCU siempre contiene un MC. La MCU puede ser llamada y el MC procesa los canales de control H.245 de todos los terminales.

Cuando dos o más terminales están en una conferencia, éstos deben usar el procedimiento de resolución maestro esclavo de la recomendación H.245 para determinar la MC que controlará la conferencia.

2.2.5 Procesador Multipunto (MP, Multipoint Processor)

A diferencia de los MC que se encargan exclusivamente del control de las conferencias, el MP recibe audio, video y/o cadenas de datos desde los terminales implicados en una conferencia multipunto centralizada o híbrida. El MP procesa estas cadenas y las devuelve a los *endpoints*.

Las comunicaciones entre el MC y el MP no están sujetas a estándar.

Un MP que procesa vídeo debe proporcionar tanto vídeo mezclado como conmutado. Si el vídeo es conmutado se enviará hacia todos los terminales de la conferencia el vídeo

proveniente de uno de los terminales. El criterio usado para hacer el cambio, y seleccionar cual es el vídeo enviado, se puede determinar a través de la detección del cambio de orador en cada momento (controlado por el nivel de audio asociado) o a través del control H.245. La mezcla de vídeo es el proceso de formatear más de una entrada de vídeo en una cadena de vídeo que el MP saca a los terminales, puede consistir en enviar hacia un mismo terminal varios cuadrantes, cada uno con el vídeo de otro terminal en la conferencia

Un MP que procesa audio debe preparar N salidas de audio de las M entradas de audio mediante conmutación, mezcla o una combinación de ambas. La mezcla de audio requiere decodificar las entradas de audio a señales lineales (PCM o análoga), realizando una combinación lineal de las señales y codificando el resultado en el formato de audio apropiado. El MP puede eliminar o atenuar algunas de las señales de entrada para reducir el ruido y otras señales no deseadas. Cada salida de audio debe tener una mezcla distinta de las señales de entrada. Los terminales deben asumir que su audio no estará presente en la cadena de audio que se les envía.

No se puede llamar a un MP, se llamará a la MCU de la que forma parte. Las entradas y salidas del MP son los canales multimedia.

2.2.6 Unidad de Control Multipunto (MCU, Multipoint Control Unit)

Es el elemento que permite soportar comunicaciones multipunto.

Una MCU es un *endpoint* que proporciona soporte para múltiples conferencias. Una MCU debe estar compuesta por un MC y cero o más MP. La MCU usa mensajes y procedimientos H.245 para implementar características similares a las enumeradas en la recomendación H.243.

Una MCU típica que soporte conferencias multipunto centralizadas estará compuesta de un MC y un MP de audio, vídeo y datos.

Una MCU típica que soporte conferencias multipunto descentralizadas estará compuesta por un MC y un MP de datos que soporte la recomendación T.120. Contará con procesamiento de audio y de vídeo descentralizado.

2.3 Arquitectura de protocolos H.323

La recomendación H.323 es considerada un paraguas de protocolos, ya que depende de otros estándares y recomendaciones para habilitar las comunicaciones multimedia en tiempo real.

La arquitectura de protocolos de H.323 se representa en la siguiente figura, incluyendo tanto el transporte de medios como el transporte de protocolos de señalización.

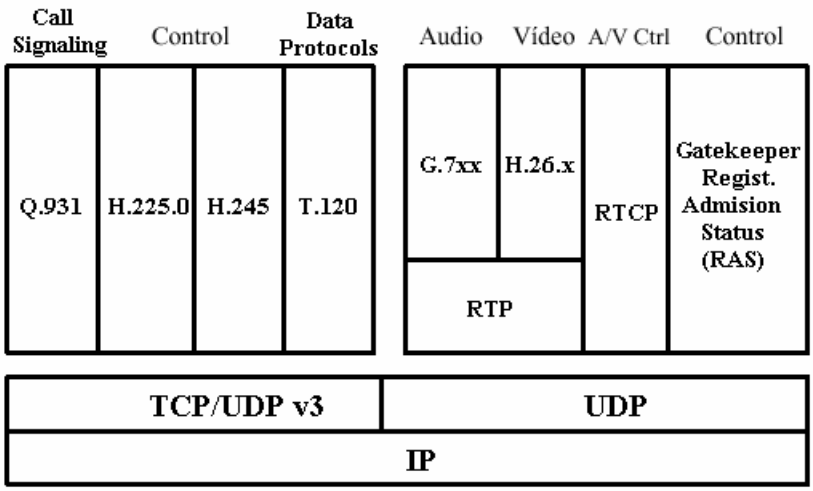

Figura 2-4. Pila de protocolos H.323

2.3.1 Protocolos de señalización

Los protocolos de señalización más importantes utilizados en el seno de H.323 son:

- Q.931: Protocolo de señalización de llamada.

- H.225.0 (*Call Signalling protocols and media stream packetization for packet-based multimedia communications*): Define la señalización entre terminales/*gateways* y *gatekeeper* (RAS). También define la señalización para establecimiento y liberación de la llamada que va por el canal de señalización de llamada. En este caso se utiliza un subconjunto de las funciones proporcionadas por Q.931.

- H.245 (*Control Protocol for Multimedia Communications*): Señalización de control extremo a extremo. La función principal es el intercambio de capacidades entre los terminales H.323 previa a la transmisión de información.

- H.235 (*Security and encryption for H-series multimedia terminals*): Trata sobre la seguridad en la comunicación incluyendo autentificación, autorización, control de llamada seguro y privacidad de los canales de voz.

- H.450 (*Call diversion suplementary service for H.323*): Señalización para el control de todos los servicios suplementarios (desvío de llamada, llamada en espera, ...).

Una llamada H.323 puede dividirse en tres fases en relación con los protocolos de señalización que intervienen en la misma:

- RAS (Registro, Autenticación y Estado): Cuando un terminal quiere hacer una llamada, pide permiso al *gatekeeper* mandando un paquete ARQ (*Admisión Request*). Este mensaje contiene, entre otras cosas, los alias del destino con quien quiere comunicarse. El *gatekeeper* puede dar permiso para la llamada con un ACF (*Admisión Confirm*) que contiene la dirección de transporte asociada al alias destino, o su propia dirección de transporte si decide encaminar la señalización H.225.0. El *gatekeeper* puede también denegar la llamada con un ARJ (*Admision Reject*) dando la razón por la cual la llamada no se ha cursado. Durante esta fase el *gatekeeper* realiza tres funciones: traducción de direcciones, autorización de llamada y gestión del ancho de banda.

- H.225.0: Es un subconjunto de mensajes del protocolo de señalización Q.931 de RDSI. Proporciona una conexión lógica entre los dos terminales. En las primeras versiones de la norma, H.225.0 se implementaba sobre TCP, pero a partir de la versión 3 existe la posibilidad de utilizar UDP por problemas de retardo.

- H.245: Tan pronto como se acaba la fase anterior los dos terminales intercambian sus capacidades y acuerdan el tipo de información que van a mandar.

Después de estas tres fases, se abren los canales lógicos entre los dos terminales según las capacidades intercambiadas y la comunicación multimedia comienza.

La mayor parte de los canales de control utilizan conexiones TCP (también UDP a partir de la versión 3), mientras que el transporte de audio y vídeo se realiza sobre UDP, por lo que pueden desordenarse, perderse o retrasarse. Por esta razón se utiliza por encima de UDP el protocolo RTP (*Real-Time Transport Protocol*).

El protocolo RTP proporciona un mecanismo para el transporte de datos en tiempo real (tales como audio y vídeo) a través de la red. Dentro de sus objetivos se utiliza para permitir compensar el *jitter* (variación de retardo) de paquetes, paquetes perdidos, el desorden de los paquetes y secuencias de errores en recepción. Como no garantiza la calidad del servicio para comunicaciones en tiempo real, el transporte de datos lo realiza bajo la supervisión del protocolo RTCP (RTP *Control Protocol*), para control del transporte en tiempo real, generar informes estadísticos entre el envío y la recepción en el protocolo RTP, indicar el estado de congestión de la red y reducir el incremento de paquetes perdidos (ajuste automático del ancho de banda). Además también permite sincronizar más de un flujo de información RTP que este protocolo, por sí mismo, no puede realizar por estar basado en la transmisión periódica de paquetes de control para todos los participantes de una sesión.

Por tanto, para establecer una comunicación H.323 se abren 3 canales de señalización (H.225.0, H.245 y RAS) más los canales lógicos de audio, vídeo y datos.

Uno de los mayores problemas de H.323 es el elevado retardo de establecimiento de llamadas. Con objeto de mejorar la eficiencia, a partir de la versión 2 de la norma se reduce el tiempo de establecimiento de llamada con dos procedimientos: *Fast Connect* y encapsulado de mensajes H.245 en mensajes Q.931.

2.3.2 Codificación de audio

La codificación es el proceso de transformación de una señal analógica a una señal digital que pueda ser transmitida por canales de ancho de banda bajo.

Los pasos que se siguen en este proceso son:

- Filtro paso banda: que limita el rango de frecuencia que se desea muestrear y reducir así los bits necesarios para digitalizar la señal analógica.
- Muestreo: convierte la señal analógica en una señal de valores discretos.
- Cuantificación: asigna un valor binario a las muestras obtenidas en la fase de muestreo.
- Codificación: existen diversas técnicas de codificación.

- PCM (*Pulse Code Modulation*) que es la modulación por pulsos codificados (básicamente es el proceso de cuantificación).

- DPCM (*Differencie Pulse Code Modulation*) que codifica la diferencia entre dos señales consecutivas.

- ADPCM (*Adaptive Differencial Code Modulation*) que se adapta dinámicamente a los diferentes tipos de señal aumentando o disminuyendo la resolución.

Los estándares especificados por la ITU-T para la codificación de audio son G.711, G.722, G.723, G.728 y G.729.

- **G.711:** Es el estándar de codificación de audio para telefonía y videotelefonía. Se basa en codificar muestras de la señal de audio a 8 KHz y asignar a esas muestras un código de 8 bits con el que conseguimos tener 256 posibles valores de la muestra con flujos de 64 Kbps. Es lo que se llama modulación por impulsos codificados (PCM). Es el estándar más apropiado para conexiones de alta velocidad.

- **G.722:** Este estándar utiliza la técnica ADCPM, es decir, no codifica el valor de la muestra, sino la diferencia con el valor anterior de la muestra, que se puede codificar con menos bits al ser una diferencia muy pequeña. Así, en este estándar se muestrea la señal a 16 KHz y se asignan códigos de 4 bits consiguiendo tener 16 posibles valores de la señal y obteniendo así mayor calidad que con el estándar G.711. Si en el anterior estándar se convertían frecuencias de 3.1 KHz a 64 Kbps, este consigue convertir frecuencias de entre 50 Hz y 7 KHz a 5.3 y 6.3 Kbps reduciendo así el empleo de ancho de banda.

- **G.723:** Al igual que G.722 comprime frecuencias comprendidas entre 50 Hz y 7KHz pero lo hace a canales de 48, 56 y 64 Kbps, consiguiendo así mayor disponibilidad y mayor calidad en la transmisión y recepción.

- **G.728:** Este estándar se basa en fórmulas matemáticas para reproducir la señal, lo que codifica son los parámetros predictores utilizados en esas fórmulas para los que sólo son necesarios 2 bits, con los que conseguimos sólo 4 niveles de cuantificación para la señal con 16 Kbps. Consigue codificar frecuencias de 3.1 KHz a flujos de 16 Kbps.

- **G.729:** Estándar equivalente a G.728 pero se reduce el régimen binario de 16 Kbps a 8 Kbps, permitiendo comprimir así los 64 Kbps.

2.3.3 Codificación de video

La técnica de codificación de vídeo más común empleada por los estándares de videoconferencia es DCT (*Discrete Cosine Transform*), la transformada discreta del coseno. Aunque existen otras técnicas de codificación de vídeo como Fractales, Cuantización de Vectores o Codificación Aritmética.

Mientras que los formatos de imagen más comunes utilizados por los diversos estándares de vídeo son CIF (*Common Intermediate Format*) que consta de 352 píxeles por línea y 288 líneas por imagen y QCIF (*Quarter Common Intermediate Format*) con 176 píxeles por línea y 144 líneas por imagen. De estos dos, el formato más utilizado es el CIF ya que ofrece una calidad de imagen mayor.

La codificación de vídeo viene especificada en los estándares H.261 y H.263.

- **H.261:** Es el estándar de vídeo común para todas las recomendaciones de videoconferencia, lo cual aumenta la interoperabilidad entre las distintas redes. Se encarga de definir el algoritmo de codificación de vídeo, el formato de las imágenes y la corrección de errores. La técnica empleada por este estándar para comprimir la información de cada fotograma es la redundancia espacial, es decir, asegura que la información correspondiente a un punto del fotograma será la misma para los puntos de alrededor, con lo que transmitiendo sólo la información de ese punto central se ahorra la de los demás. Mientras que la técnica empleada para comprimir una secuencia de fotogramas es la redundancia temporal, basada en transmitir sólo la diferencia entre un fotograma y el siguiente en cuanto que la diferencia entre ambas imágenes será mínima. Este estándar se basa en una codificación de vídeo para velocidades entre 40 Kbps y 2 Mbps. Además, debe soportar obligatoriamente el formato de imagen QCIF, mientras que el formato CIF es opcional.

- **H.263:** Este estándar ofrece mejoras respecto a H.261 desde dos aspectos:

  - Soporta más formatos de imagen, como son: 16CIF, 4CIF, CIF, QCIF y Sub-QCIF (para transmisiones en Internet de baja velocidad como módems de 28.8 Kbps).

- Mejora la técnica de redundancia temporal, ya que tiene en cuenta no sólo los fotogramas pasados sino también los siguientes esperados, y además ofrece mayor calidad al ampliar la zona en la que busca el macrobloque en la imagen siguiente a 32 puntos en lugar de los 16 que usa H.261.

De forma que para una determinada velocidad de transferencia, H.263 ofrece mayor calidad de imagen que H.261 con resoluciones que van desde Sub-QCIF a 4xFCIF.

2.3.4 Datos

Se requiere la capacidad de intercambiar datos en tiempo real para actividades tales como compartir aplicaciones, transferencia de ficheros, transmisión de fax y mensajes inmediatos. La recomendación T.120 proporciona esta capacidad opcional a H.323.

T.120 es un protocolo de comunicación de datos en tiempo real diseñado específicamente para las necesidades de conferencia. Al igual que H.323, la recomendación T.120 es un paraguas de un conjunto de estándares para permitir el intercambio en tiempo real de datos específicos entre varios clientes sobre diferentes redes. T.120 proporciona varias ventajas sobre la transmisión de datos habitual, tales como:

- Soporte para conferencia multipunto: T.120 permite entrega de datos multipunto lo que habilita actividades de colaboración en grupo. La MCU maneja la mezcla o la conmutación de datos de forma similar al vídeo y audio.

- Independencia de plataforma y red: T.120 opera en lo alto de la capa de transporte de la red, de esta forma es independiente del hardware y software de la red.

- Interoperatibilidad: Se referencia a T.120 en todos los estándares de conferencia H.323, de esta forma y junto con la independencia de la plataforma, asegura un alto grado de interoperatibilidad en el nivel de aplicación.

- Soporte de envío múltiple: T.120 soporta el envío múltiple de cadenas de datos en las redes que lo permiten. Es flexible con la mezcla de envío múltiple o único, que también es posible durante una conferencia.

- Otros beneficios: T.120 proporciona capacidad de corrección de errores en lo alto de la capa de transporte de la red, asegurando así entrega fiable. En general, T.120 tiene

una arquitectura escalable y ampliable preparada para añadir nuevas aplicaciones que mejoren la entrega de datos de forma eficiente y fiable en tiempo real sobre un grupo de usuarios.

La siguiente figura muestra la pila de protocolos T.120:

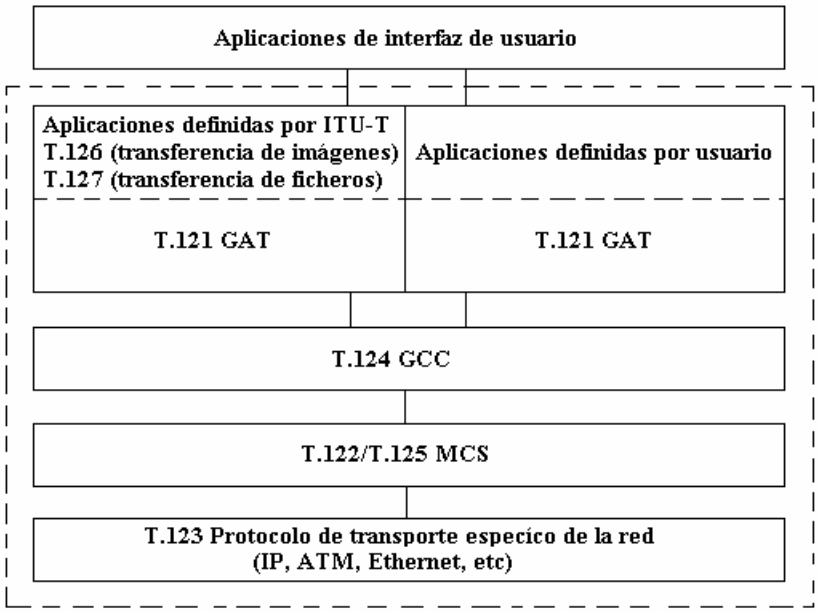

Figura 2-5 Pila de protocolos T-120

- T.123: protocolo de transporte para conferencia de datos creado después de la capa de transporte de OSI (*Open Systems Interconnection*). La implementación de esta capa se adapta a las capacidades de la red sobre la que opera, sea ATM, IP, etc. T.123 también proporciona soporte de detección y corrección de errores a las comunicaciones T.120.

- T.122 y T.125 (MCS, *Multipoint Communication Service*): T.122 especifica los servicios, mientras que T.125 define la implementación del protocolo. El principal objetivo de MCS es la coordinación y la prioridad de los datos sobre múltiples conferencias.

- T.124 (GCC, *Generic Conference Control*): facilita la creación y gestión de conferencias con características tales como creación de conferencia, entrada y salida de terminales de una conferencia, seguridad de autenticación, gestión de los recursos del servicio y gestión de la información.

- T.121 (GAT, *Generic Application Template*): especifica una plantilla para el desarrollo de aplicaciones de conferencia de datos. Siguiendo esta plantilla de desarrollo o proyecto se asegura la compatibilidad de aplicaciones que usan el GCC y el MCS. El uso de la plantilla es obligatorio para las aplicaciones estándar y es muy recomendable para las aplicaciones definidas por el usuario. Las aplicaciones definidas por la ITU-T incluyen T.126 para el intercambio de imágenes y T.127 para transferencia de ficheros binarios multipunto.

Capítulo 3: Marco conceptual

3.1 Introducción

Este trabajo surgió de la necesidad de incorporar una MCU H.323 al proyecto CIMA (Centro de Información Multimedia Avanzado), para lo cual se parte del código desarrollado por *OpenH323 Project*. En este capítulo se explican brevemente dichos proyectos y cómo es la comunicación de la MCU con CIMA, así como con el resto de los programas con los que se relaciona.

3.2 Proyecto CIMA

El principal objetivo de este proyecto era integrarlo dentro de la arquitectura del proyecto CIMA, que es un sistema proveedor de servicios de teleasistencia sobre IP. Este sistema posee elementos de gestión (ACD o *Automatic Call Distributor*) encargados de gestionar y distribuir las llamadas que los clientes realizan hacia ciertos asesores, los cuales están asignados a un determinado perfil de asistencia. Las peticiones de los clientes son enviadas a través de una simple página Web, mediante un interfaz de tipo "*clic to call*" (pulsar para llamar). La elección del asesor se realiza de acuerdo a información relativa a esta página Web así como a datos adicionales extraídos del cliente. Las llamadas multimedia se realizan sobre IP con el conjunto de estándares H.323.

3.2.1 Arquitectura del sistema CIMA

Se puede representar esquemáticamente la arquitectura del proyecto CIMA según la figura 3-1.

Los elementos que se interrelacionan en ella son:

- **Cliente:** Terminal con capacidad de navegar por internet (mediante Internet Explorer) y establecer comunicaciones H.323 en modo dúplex para audio y en modo simplex (sólo recepción) para vídeo. Para ello se exige que tanto cliente como asesor incorporen el software *NetMeeting* de Microsoft.

- **Asesor:** Terminal identificable por un alias con capacidad de videoconferencia H.323 en modo *full-dúplex*. Cada asesor debe pertenecer al menos a un perfil de trabajo, el

cual identifica el servicio de teleasistencia. Es el encargado de resolver las dudas que le planteen los clientes.

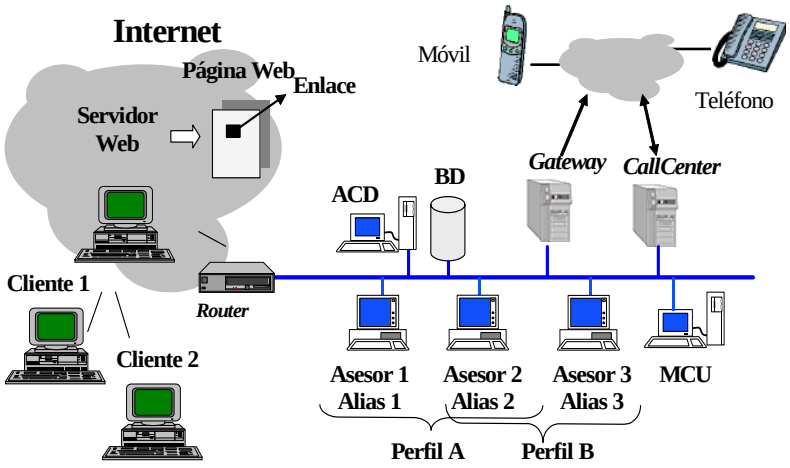

Figura 3-1. Arquitectura del proyecto CIMA

- **Servidor Web:** Simple servidor de páginas HTML. Contiene las páginas que dan acceso al sistema. En la programación del servidor se han distinguido dos áreas: por un lado se ubican las páginas de acceso de los clientes modificadas para incorporar la funcionalidad del sistema CIMA. Por otro lado, se genera una zona segura para controlar el acceso de los asesores con protocolo HTTPS y petición de certificado personal.

- **ACD *(Automatic Call Distributor)*:** Es el núcleo del sistema. Se encarga de seleccionar los asesores que se ajusten a los perfiles demandados por los clientes. Esta decisión se fundamenta en el análisis de parámetros como la dirección IP del cliente, el botón y la página *Web* desde la que se realizó la petición. Una vez estudiados, el ACD deduce el perfil requerido para la teleasistencia y escoge a uno de los asesores que se encuentre disponible y que sea experto en dicho perfil. Según la configuración de cada perfil, el ACD puede elegir asesores que se encuentren sin ofrecer servicio o bien unir al cliente a una multiconferencia en la que se esté tratando el tema que él solicita.

- **MCU (*MultiPoint Control Unit*):** Es el elemento del sistema encargado de la gestión de los flujos de audio y vídeo. Permite la creación de salas de multiconferencia en las

que un asesor puede atender a varios clientes simultáneamente. Al actuar la MCU como puente entre clientes y asesores, se le añade al sistema una componente de seguridad ya que los flujos de audio y vídeo pasan siempre a través de él. De esta manera, se evita que se establezca una llamada directa entre el cliente y el asesor.

- ***Gateway* H.323:** Este componente opcional en el sistema permite realizar una llamada de audio desde un cliente H.323 hacia un teléfono, actuando como pasarela entre IP y la red telefónica clásica. El escenario de utilización del *gateway* se limita al paso de llamada atendida por un asesor a otro asesor que se encuentra disponible a través de un teléfono fijo o de un teléfono móvil.

- ***Call Center* telefónico**: Existe la posibilidad de implantar el sistema en un entorno en el que ya se esté ofreciendo un servicio de asistencia telefónica clásico. CIMA es capaz de interactuar con centralitas telefónicas clásicas de manera que no se permite que un asesor reciba llamadas simultáneas a través del *call center* telefónico y a través de CIMA.

3.2.2 Funcionamiento básico sin MCU

La sesión de teleasistencia se dispara cuando un cliente accede a la página correspondiente del Servidor Web y presiona en el hiperenlace de solicitud a un ACD. En la figura 3.2 se esboza, a grandes rasgos, el flujo de información entre los elementos a la hora de establecer la sesión de teleasistencia.

En función de la información extraída al cliente (por ejemplo su dirección IP) y de la página desde la que se solicitó el servicio, el ACD acepta (o no) la llamada, concediendo un tipo concreto de teleasistencia (perfil) al cliente. En función de este perfil y de la disponibilidad de los asesores asignados al mismo, el ACD designa al asesor que dará servicio al cliente, trasladándole la información que conoce del mismo (por ejemplo: página desde la que llama, navegador que emplea, información sobre sus últimos accesos al servicio de teleasistencia, ...). Simultáneamente el ACD dará orden al cliente de que arranque una sesión H.323 con el asesor elegido.

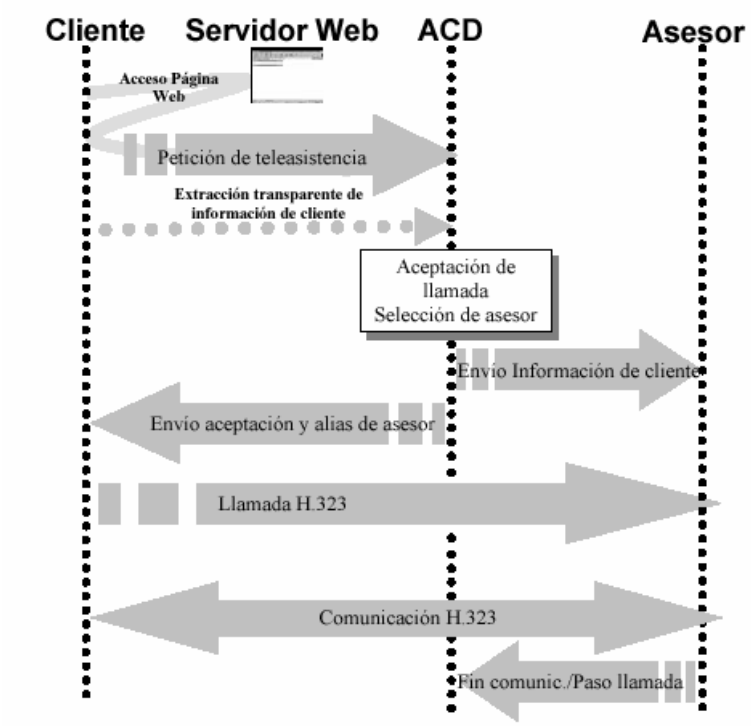

Figura 3-2. Interacción de los elementos en CIMA

3.2.3 Comunicación con la MCU

Entre las líneas de desarrollo a seguir del proyecto CIMA ya se había propuesto añadir un elemento más a la arquitectura, una MCU H.323 que permitiese el establecimiento de sesiones multipunto entre varios asesores y uno o varios clientes. La MCU permite que varios usuarios se comuniquen simultáneamente en una habitación virtual de multiconferencia, pudiendo gestionar a la vez distintas habitaciones.

La comunicación entre el ACD y la MCU será mínima, y se hará mediante ficheros. Básicamente habrá dos tipos de ficheros: ficheros de acceso y ficheros de eliminación. La MCU no permitirá establecer una videoconferencia a ningún cliente o asesor que no tenga su correspondiente fichero de acceso creado por el ACD. Cuando el ACD quiere que un cliente o un asesor terminen la comunicación creará el correspondiente fichero de eliminación.

El flujo de información actual entre los elementos de la arquitectura, ya con la MCU, se podría esquematizar como muestra la siguiente figura:

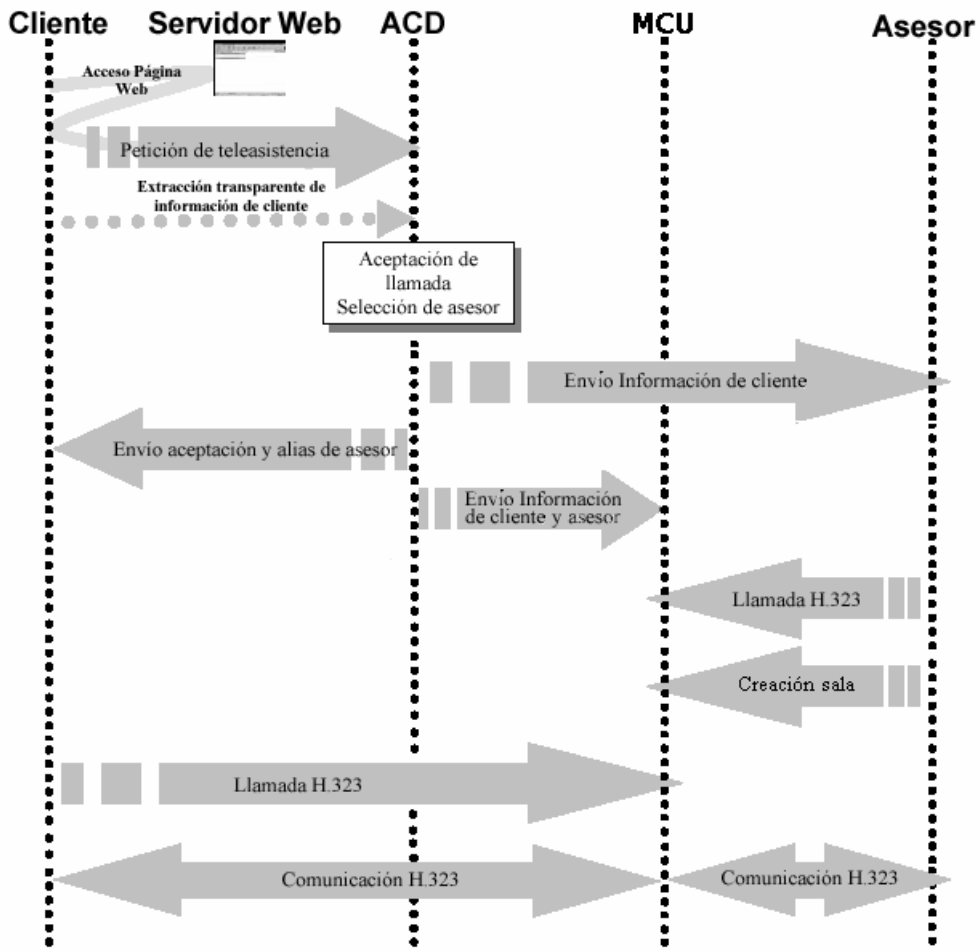

Figura 3-3. Interacción de los elementos en CIMA con MCU

Inicialmente el asesor debe acceder al sistema. Para ello se conecta a una página *Web* segura en la que se le exige su nombre de usuario y su clave así como un certificado digital.

El cliente visita una página *web* en la que se encuentran uno o varios enlaces que permiten el acceso al sistema CIMA. Cuando pulsa sobre uno de ellos, se inicia la petición de teleasistencia al ACD, que extraerá los parámetros del enlace e información del cliente. Estos datos son utilizados para seleccionar el asesor disponible que atenderá la llamada según la configuración almacenada en la base de datos. Una vez seleccionado, el ACD muestra toda la información referente al cliente en el interfaz del asesor asignado e informa a la MCU para que genere una sala de multiconferencia indicándole los participantes (mediante los ficheros de acceso).

Si el asesor acepta la llamada, tanto él como el cliente establecen una comunicación H.323 con la MCU, que se encarga de intercambiar los flujos de audio y de vídeo entre ambos.

Cuando se conecta a la MCU un asesor se crea una sala, cuyo nombre será el del propio asesor, en la que éste presta el servicio al cliente que el ACD le asigne. Cada multiconferencia se asocia a una sala o habitación.

Al integrar la MCU en el sistema se deben añadir a las funcionalidades del ACD las siguientes tareas:

- Creación de los ficheros de acceso y eliminación: el ACD será el encargado de crear los ficheros según el formato que se explicará en la siguiente sección.

- Análisis y detección en los clientes y asesores del codec MS-GSM, y, en caso necesario, la posibilidad de cargar dicho codec si el cliente o asesor no dispone de él.

La MCU puede funcionar de forma independiente, sin integrarse en el proyecto CIMA, pero habrá que crear los ficheros de forma manual para cada usuario que se desee conectar. Del mismo modo habrá que cargar el codec MS-GSM al *NetMeeting* del usuario, en caso de que no disponga de él, si la características de su conexión así lo requieren. En el siguiente capítulo se explicará la necesidad de este codec.

3.2.4 Ficheros

La comunicación entre el ACD y la MCU básicamente se hará mediante dos tipos de ficheros en los que indicará cómo se conectará cada usuario y cuándo desconectar a alguien. Estos ficheros son:

3.2.4.1 Ficheros de acceso

Existirá un fichero de acceso por cada cliente o asesor que se desee conectar a la MCU. Este fichero tendrá por nombre el nombre de dicho cliente o asesor, según lo tengan estos configurado en *NetMeeting* (dentro del menú Herramientas, en Opciones).

Dentro de dicho fichero existirán tres líneas:

- 1ª línea: Nombre de la habitación en la que dicho usuario entrará.

- 2ª línea: 0 = modo punto a punto, 1 = modo multipunto.

- 3ª línea: 0 = no trabaja con vídeo, 1 = si trabaja con vídeo.

En el siguiente capítulo se verá con más detalle la razón de cada una de estas líneas para el correcto funcionamiento de la MCU, según las necesidades de cada usuario.

Cuando el usuario sea un asesor, el nombre de la habitación se corresponderá con el nombre del usuario (y del fichero), en caso de ser un cliente el nombre de la habitación nunca será igual al nombre del usuario. De esta forma la MCU podrá distinguir entre asesores y clientes cuando lo necesite.

3.2.4.2 Ficheros de eliminación

Cuando el ACD quiere indicar a la MCU que un usuario (cliente o asesor) debe terminar su videoconferencia, crea un fichero de eliminación cuyo nombre será E_[*nombre*], siendo *nombre* el que corresponda a la configuración del *NetMeeting* del usuario igual que en los ficheros de acceso.

Dentro de este fichero sólo existirá una línea que indica la habitación en la que se encuentra el cliente o el asesor.

3.3 OpenH323 Project

Para el desarrollo de la MCU se partió del código creado por el *OpenH323 Project* [1], que es un esfuerzo de la compañía australiana *Equivalence Pty Itd* [10] para generar código que permita la implementación del protocolo ITU-H323, para desarrolladores a nivel comercial o personal, el código se encuentra bajo la licencia MPL (*Mozilla Public Licence*) [13]. Es un código en lenguaje C++, el cual implementa las clases necesarias para desarrollar programas de teleconferencia basados en este estándar.

OpenH323 Project ya implementa una MCU, llamada *OpenMCU*, para hacer uso de ella son necesarias dos librerías, por un lado *PWLib* y por otro *OpenH323*.

OpenH323 se encuentra en constante desarrollo y actualización, para el desarrollo de nuestro proyecto partimos de las siguientes versiones:

- PWLib v1.4.11
- OpenH323 v1.11.7

- OpenMCU v.1.1.7

3.3.1 Librería PWLib

PWLib (*Portable Windows Library*), es una librería de clases que se desarrolló con el objetivo de proporcionar las abstracciones del sistema operativo necesarias para poder crear aplicaciones que pudieran funcionar tanto en Microsoft Windows como en Unix, sin modificar el código fuente.

Esta librería contiene clases que encapsulan funcionalidades de I/O, GUI, multi-*threading* y red, y también clases que representan clases de contenedores básicos, tales como arrays, listas, y diccionarios.

3.3.2 Librería OpenH323

Es la librería central de *OpenH323 Project*, crea una implementación operativa del protocolo de videoconferencia H.323.

Las clases de *OpenH323* no hacen uso directo de las librerías dependientes del sistema operativo, estas llamadas las hacen a la librería *PWLib*.

En el Apéndice A se esboza brevemente la arquitectura de dicha librería centrándose en sus clases principales.

3.3.3 OpenMCU

Es el código de la MCU desarrollada por *OpenH323 Project* y en el cual nos basamos para realizar nuestro proyecto. En el siguiente capítulo se explicarán con detalle las modificaciones realizadas a este código.

Entre sus características podemos citar:

- No necesita para funcionar añadir los codec en hardware.
- Soporta los codecs de audio: G.711, LPC-10, GSM y MS-GSM.
- Soporta los codecs de vídeo H.261.

- Pueden darse al mismo tiempo diferentes conferencias gracias al uso de habitaciones.

- Muestra información de las llamadas en progreso.

Básicamente OpenMCU trabaja colocando un proceso *listener* H.323, y espera llamadas entrantes. Cuando llega una llamada se determina a qué conferencia la une según la habitación. Las habitaciones se crean automáticamente.

Aunque el código es libre para su uso y modificación, la documentación que existe en la página web de *OpenH323 Project* es escasa para los desarrolladores, por lo que para la realización de este proyecto fue necesario hacer previamente una labor de ingeniería inversa para tratar de comprender la estructura del código y el funcionamiento del mismo.

Mediante la herramienta CASE *Rational Rose C++ Analyzer 4.0* y aplicando la ingeniería inversa se construyeron los modelos de objetos que facilitaron la comprensión de la jerarquía de clases, así como las relaciones que se establecen entre ellas, sus métodos, etc.

El Apéndice B se centra en la arquitectura de la MCU.

3.4 WatchDog

Otro programa que también se desarrolló como parte del proyecto, aunque no se fijó en los objetivos que se marcaron en el anteproyecto, es el *WatchDog*.

Su necesidad se vio al realizar las primeras pruebas según se iba modificando la MCU. Se observó que había algunas caídas del sistema sin explicación exacta, ya que eran aleatorias y escasas. Se decidió entonces crear un programa que se encargara de arrancar la MCU y, en caso de caída de ésta, reiniciarla automáticamente.

Es un programa escrito en C que mediante la llamada al sistema con `fork()` se crea un proceso hijo que será la MCU. Con su PID (identificador del proceso), se podrá saber si el proceso hijo ha caído o no, y, en caso necesario, reiniciarlo.

Este programa ha facilitado la realización de las pruebas, ya que ha evitado tener que estar pendiente del ordenador de la MCU mientras se modifican las opciones de *NetMeeting*, en los ordenadores clientes, para probar los distintos escenarios que se comentarán en la sección de pruebas (4.5).

Además, para tener más información, este programa registra en un fichero la fecha y hora en la que se arranca la MCU, así como la de cada caída y reinicio de ésta, y por último también registra la de finalización correcta del programa.

De esta forma se observó que dichas caídas desaparecieron según se iban corrigiendo las modificaciones que se hacían al código.

Capítulo 4: Desarrollo del Proyecto

4.1 Introducción

Como ya se ha visto en el capítulo anterior el principal objetivo de este trabajo es integrarlo en el proyecto CIMA. Esta adaptación supone una serie de modificaciones al código *OpenMCU* que se explicarán con detalle en este capítulo.

La dificultad no ha estado tanto en las modificaciones en sí, sino en la labor de ingeniería inversa previa que se tuvo que realizar para poder saber dónde era más conveniente realizar cada modificación, aprovechando el código que ya existía, de forma que las soluciones encontradas fueran lo más óptimas posibles.

4.2 Compilación del código fuente

El primer paso fue la compilación del código fuente de las librerías *PWLib* y *OpenH323*, para poder, a continuación, compilar *OpenMCU*.

Una vez bajados los ficheros comprimidos de la página de *OpenH323 Project* se extraen los fuentes, cada uno en su directorio correspondiente (pwlib, openh323, openmcu).

Antes de compilar hay que definir las siguientes variables de entorno:

```
PWLIBDIR=$HOME/…/pwlib
export PWLIBDIR
OPENH323DIR=$HOME/…/openh323
export OPENH323DIR
LD_LIBRARY_PATH=$PWLIBDIR/lib:$OPENH323DIR/lib
export LD_LIBRARY_PATH
```

según los directorios en los que se hayan extraído los fuente. Después se realiza la compilación en el siguiente orden, primero la librería *PWLib*:

```
cd $PWLIBDIR
./configure
make
```

después la librería *OpenH323*:

```
cd $OPENH323DIR
make opt
```

por último, para compilar *OpenMCU*:

```
cd $HOME/.../openmcu
make opt
```

4.3 Modificaciones

Una vez probado el código de *OpenMCU*, y visto lo que hacía y lo que no, se añadieron nuevas funcionalidades al programa, según se explica a continuación. Estas modificaciones las podemos agrupar en las siguientes categorías:

1. Argumentos/Opciones de configuración.
2. Control de acceso.
3. Desconexión de usuarios.
4. Vídeo.
5. Consultas.
6. Codec audio.

Del grupo 1 al 5 sólo hubo que modificar los ficheros *main.cxx* y *main.h* de *OpenMCU*, sin embargo para el punto 6 se tuvieron que modificar algunos ficheros de la librería *OpenH323*, tal y como se explicará en la sección 4.3.6.

4.3.1 Argumentos/Opciones de configuración

OpenMCU recibe en la línea de comandos las opciones que determinarán su funcionamiento. Sin embargo nuestra MCU lee esas opciones directamente de un fichero, de forma que se establezcan fácil y rápidamente las características deseadas, y además queden fijadas de una vez a otra, pero pudiéndose modificar fácilmente.

El fichero se denominará *opciones* y deberá estar en el mismo directorio desde el que se ejecuta la MCU. Dentro sólo tendrá una línea que será la cadena de opciones, tal y como éstas se escribirían en la línea de comandos al ejecutar *OpenMCU*.

Para recoger las opciones *OpenMCU* definía la variable `args` y la inicializaba según:

```
PConfigArgs args(GetArguments());
```

dentro de la función `void OpenMcu::Main()`. Para conseguir que las opciones las tome de un fichero, simplemente leemos ese fichero y almacenamos la cadena que contiene en la variable: `char opciones[80]`, de forma que ahora la variable `args` se inicializa ahora según:

```
PConfigArgs args(opciones);
```

Si no encuentra, o no existe el fichero, se han fijado en la MCU unas opciones por defecto (se inicializa `opciones` con `"-n –v"`), que indican que va a trabajar sin *gatekeeper* y con vídeo.

Por otro lado se han añadido nuevas opciones a la MCU, como son:

- `-P --prefer codec`: modifica el orden de preferencia de los codecs locales de la MCU colocando a *codec* en primer lugar. Se puede usar varias veces.
- `-D -–disable codec`: elimina *codec* de la tabla de codecs locales de la MCU.

Dentro de *OpenMCU* se cargan los codecs en la tabla local, y en función del orden en el que se hacía, se definía el orden de preferencia de los mismos. Estas opciones nos permiten alterar la tabla de codec local de la MCU, cambiando el orden de prioridad o eliminando codecs, pudiendo así realizar distintas pruebas sin tener que modificar el código.

Como se explicará en la sección codec de audio (4.3.6), se definieron dos tablas de codecs locales, punto a punto y multipunto. Estas dos opciones se aplicarán únicamente a la tabla de codec multipunto de la MCU.

Para añadir opciones a la MCU primero hubo que añadirlas al conjunto de opciones posibles, también dentro de la función `void OpenMcu::Main()`.

```
args.Parse(
        …
        "P-prefer:"
```

```
"D-disable:"
…
, FALSE);
```

y después crear el código :

```
// Cambio del orden de preferencia de los codec
if (args.HasOption('P'))
endpoint.ReorderCapabilities(args.GetOptionString('P').Lines());
// Borrado de codec
if (args.HasOption('D'))
endpoint.RemoveCapabilities(args.GetOptionString('D').Lines());
```

Tanto `ReorderCapabilities` como `RemoveCapabilities` estaban ya implementadas en la clase *H323EndPoint* de la librería *OpenH323*, aunque no se llegaban a usar en *OpenMCU*.

Por otro lado nuestro programa sí recibe un argumento: la ruta en la que tendrá que buscar tanto los ficheros de acceso como los ficheros de desconexión de los usuarios. Para almacenar el directorio concreto definimos la variable global dentro del fichero *main.cxx*:

```
PString RUTA_FICHEROS = ".";
```

Y ya dentro de la función `void OpenMcu::Main()` se recoge primero el valor del argumento:

```
PConfigArgs ruta(GetArguments());
```

Después se le añade el carácter "`/`" al final si no lo tiene, y se le asigna a la variable `RUTA_FICHEROS` para su uso.

4.3.2 Control de acceso

En *OpenMCU* las nuevas habitaciones se crean automáticamente y hay una habitación por defecto, llamada "*room101*", para todos los clientes H.323 que no especifiquen o no puedan especificar una, como es el caso de los usuarios que usan *NetMeeting*. Esto implica que todos los que llamen a través de *NetMeeting* se conectan a esta habitación y, por tanto, no se podrían tener distintas habitaciones simultáneamente.

Para solucionar este problema se decidió trabajar con unos ficheros de acceso, en los que se indica, entre otras cosas, qué habitación se asocia a cada usuario. La MCU busca estos ficheros en la ruta que se le pasa como argumento al programa.

Para implementar este control de acceso se ha modificado la función:

```
H323Connection::AnswerCallResponse
   MyH323Connection::OnAnswerCall (const PString & caller,
                                   const H323SignalPDU & setupPDU,
                                   H323SignalPDU & /*connectPDU*/)
```

Se puede ver que entre los parámetros que recibe esta función está `caller`, que almacena la cadena que identifica al usuario que está llamando, de esta cadena se extrae el nombre, que será el mismo para su fichero de conexión.

Se lee este fichero y se inicializa la variable `roomID` (de la instancia de la clase *MyH323Connection* que se le asocia al usuario), al valor que esté en la primera línea del fichero.

Si un usuario intenta conectarse a la MCU y no se ha generado un fichero con su nombre, o no se tiene acceso a él, no se le permite el acceso a ninguna videoconferencia, con lo que nunca llegará a usarse la habitación por defecto.

4.3.3 Desconexión

Existen varios casos en los que se desconectará a un usuario de la MCU que no estaban contemplados en *OpenMCU*. Estos son:

4.3.3.1 Tiempo máximo de espera

Cuando alguien intenta establecer una videoconferencia se tendrá un tiempo máximo de espera (que se ha establecido en un minuto) desde que se crea la habitación hasta que otro usuario se conecta a la misma habitación. Si pasa este tiempo y nadie más se ha conectado, se desconectará al usuario y se borrará la habitación.

Para poder implementar este requisito se tiene que comprobar continuamente si existe alguna habitación con una única conexión, y el tiempo que ésta lleva abierta.

Se ha modificado en este caso la función:

```
void MyH323EndPoint::AwaitTermination()
```

ya que aquí se comprueba cada segundo los posibles cambios del sistema.

En ella se define un intervalo de tiempo de un minuto (60 segundos), usando en este caso una clase de la librería *PWLib*:

```
PTimeInterval *tespera = new PTimeInterval(0,60,0,0,0);
```

Por otro lado se puede saber cuándo se conectó un usuario gracias a la función `GetConnectionStartTime()` de la clase `H323Connection`. Después de localizar la conexión exacta, si se almacena este valor en la variable `callStart,` esta modificación queda resumida en la línea:

```
if ((tespera->Compare(now-callStart))<0)  conn->ClearCall();
```

ya que la función `ClearCall` elimina una conexión concreta.

4.3.3.2 Ficheros de desconexión

Como ya se ha comentado, el programa recibe como parámetro una ruta de ficheros. Si el ACD desea que se desconecte a un usuario concreto, debe crear en ese directorio un fichero cuyo nombre sea E_[nombre_usuario], y que contenga el nombre de la habitación a la cual está conectado el usuario.

La MCU comprueba continuamente si existe algún fichero en ese directorio que comience por E_, en cuyo caso procede a desconectar al usuario de la sala o videoconferencia en la que se encuentre. Se borrará el fichero inmediatamente después.

De nuevo se modifica la función:

```
void MyH323EndPoint::AwaitTermination()
```

así se comprueba cada segundo si se ha creado algún fichero con esas características en el directorio indicado.

Se incluye en este caso la librería de C++: `#include <dirent.h>`, para el manejo de directorios, y poder definir así las variables: `DIR *dirp;` y `struct dirent *dp;`, de forma que `dirp` está asociado al directorio y `dp` es una estructura mediante la cual se pueden obtener los nombres de los ficheros de ese directorio.

Una vez localizado un fichero con esas características se extrae, de su nombre, el nombre del usuario a desconectar, después se busca la conexión concreta y se borra.

4.3.3.3 Se va el asesor

Como ya se ha explicado, el programa distingue entre asesores y clientes. En cada sala existe un asesor que da soporte a los usuarios en la misma habitación. Cuando un asesor se desconecte se tendrá que desconectar a todos los clientes que estuvieran en la misma multiconferencia que él.

En este caso se ha modificado la función:

```
void MyH323EndPoint::RemoveMember(MyH323Connection * oldConn)
```

en la que se siguen los pasos para desconectar a un usuario, se añade aquí la comprobación de si es un asesor el que se va, en cuyo caso se localizan las demás conexiones de la habitación en la que estaba, se cerrarán todas y se borrará la habitación.

4.3.3.4 Sólo queda una conexión en una habitación

Cuando sólo hay dos usuarios en una habitación y uno de ellos se va, entonces hay que desconectar también al otro usuario, aunque éste no lo haya solicitado.

Se modifica de nuevo la función:

```
void MyH323EndPoint::RemoveMember(MyH323Connection * oldConn)
```

Dónde se comprueba ahora si sólo queda un usuario en la habitación de la cual se acaba de desconectar alguien, en este caso se cierra también su conexión, y se borra la habitación.

En un principio este caso era más complicado, ya que cuando quedaban sólo dos personas en una habitación y una de ellas cerraba su conexión, se comprobaba el código apuntado en una línea del fichero de acceso del usuario que quedaba, de forma que si era un 0, automáticamente se desconectaba al usuario y se destruía la habitación, no permitiendo así que se quedara esperando hasta que se conecte otro usuario a su videoconferencia. Sin embargo, si era un 1 sí se permitía que esperara un tiempo. Esta característica se eliminó de la MCU al considerarse que ya no era necesaria, y se simplificó este caso hasta lo explicado anteriormente.

4.3.4 Vídeo

OpenMCU muestra hasta 4 vídeos simultáneos por habitación, esto es, divide el vídeo en cuatro cuadrantes que asocia a cuatro de los usuarios que haya conectados en cada habitación.

Nuestra MCU sólo muestra (envía) el vídeo de un usuario si éste es un asesor, esto es, los clientes verán al asesor, pero el asesor no verá a los clientes. Esto es así por requisitos de diseño del proyecto CIMA.

Se modifica ahora la función:

```
BOOL MyH323EndPoint::AddVideoPosnToken
        (const PString & thisToken, PString roomID)
```

en la que sólo se asociará una posición de vídeo si es un asesor el usuario que se ha conectado.

Se han implementado las funciones BOOL `GetEsAsesor()` y `void SetEsAsesor(BOOL a)` dentro de la clase *MyH323Connection* para saber en cada momento si una conexión está asociada o no a un asesor.

4.3.5 Consultas

Además de los datos que *OpenMCU* ofrece para su consulta cuando hay usuarios conectados, se estableció que se guardaran también tanto la hora de creación de cada una de las habitaciones como la hora a la que se desconectó el último usuario de cada una, si es que alguno lo hizo, y que ambas se pudieran consultar en cualquier momento.

Se creó entonces, dentro de la clase *MyH323EndPoint*, la propiedad:

```
StringListDict roomTiempos;
```

Usando una clase definida en la librería *PWLib*, que nos ofrece una lista en la que cada campo es otra lista. De esta forma la primera lista contiene los identificadores de cada habitación, y para cada uno se almacena, en su lista correspondiente, en el primer campo el tiempo de creación de la sala, y en el segundo cuándo se ha desconectado el último usuario de la misma. Este segundo campo sólo se rellena cuando alguien se desconecta, y se va actualizando según se vayan desconectando los usuarios de esa habitación.

Esta estructura permite almacenar fácilmente más datos relativos a las habitaciones, si fuese necesario.

También se definió el método de consulta:

```
StringListDict GetRoomTiempos(){ return roomTiempos;}
```

Y se incorporó al menú de la MCU la opción “t” con la que se listaba estos datos, de esta forma se puede consultar en cualquier momento esta información de las habitaciones que existan en ese instante.

Se modifica `roomTiempos` cada vez que se crea una habitación, en:

```
void MyH323EndPoint::AddMember(MyH323Connection * newMember)
```

o cada vez que alguien se desconecta, en:

```
void MyH323EndPoint::RemoveMember(MyH323Connection * oldConn)
```

Aquí también se borra la lista correspondiente cuando se borra una habitación.

Además se han añadido otras posibles consultas al menú de la MCU, de esta forma la opción “i” sólo muestra una lista de las conexiones en cada habitación y el modo de conexión de cada una (punto a punto o multipunto), simplificando la información que proporciona la opción “s” y facilitando su visualización en pantalla. Por otro lado la opción “p” lista todos los codecs remotos para cada conexión.

4.3.6 Codec de audio

Hasta ahora todas las modificaciones se han hecho en los ficheros *main.cxx* y *main.h* de *OpenMCU*, con lo cual no se había necesitado estudiar muy en profundidad las librerías que ésta necesitaba, aunque sí se consultó qué clases de *PWLib* se podían utilizar, y cómo se definían en *OpenH323* las clases de las que heredaban las principales de la MCU.

Fue debido a los problemas con los codec de audio lo que obligó a tener que realizar un estudio más detallado de la librería *OpenH323*, aunque no tanto como para generar una documentación robusta de ésta, sí lo suficiente para poder modificar el código ajustándolo a las necesidades.

El problema más difícil de solucionar se encontró realizando las pruebas, y no era un problema en sí del código de la MCU, sino de los codec de audio de que ésta disponía, ya que

para los distintos anchos de banda de los que dispone cada usuario, dependen los codec de audio que éstos pueden usar realmente.

En *NetMeeting*, cada usuario puede configurar el ancho de banda en el menú Herramientas > Opciones > General > Configuración del ancho de banda:

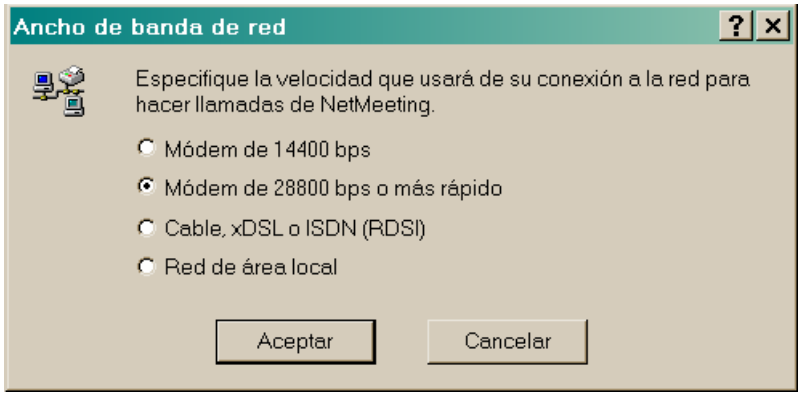

Figura 4-1. Configuración del ancho de banda en NetMeeting

Cuando se seleccionan una de las dos primeras opciones sólo se disponen de los siguientes codecs:

- Microsoft G.723.1, 8KHz Mono, 6400 Bits/s
- Microsoft G.723.1, 8KHz Mono, 5333 Bits/s

Sin embargo en los dos siguientes ya se dispone además de:

- Ley u de CCITT, 8000KHz; 8 bit; Mono
- Ley A de CCITT, 8000KHz; 8bit; Mono

Se observó en las pruebas que cuando los usuarios se conectaban mediante un módem a la MCU no se podían llevar a cabo las conversaciones debido a que la MCU no dispone de un codec de audio apropiado. *NetMeeting* tiene asociado el codec G.723.1 para estos casos, sin embargo no existe una licencia de libre uso de dicho codec y, por tanto, la MCU no lo implementa.

Se estudiaron entonces varias soluciones según se explica a continuación.

4.3.6.1 Codec GSM

La primera solución que se estudió, y que se proponía en la página de *OpenH323 Project*, fue usar el codec GSM, que sí es de libre uso y, aunque *NetMeeting* no lo incluye en

un principio, sí existe un programa (*instcodec.exe*) por el que se le puede añadir. Por otro lado la MCU tiene implementado tanto la versión para *Microsoft* y *NetMeeting* (MS-GSM), como para el resto de los programas clientes (GSM-06.10), por lo que no había que hacer ninguna modificación al código.

Sin embargo cuando se realizaron las pruebas correspondientes se observó que se introducían ruidos en las conversaciones que dificultaban el correcto entendimiento entre los usuarios, estos ruidos se incrementaban según aumentaba el número de usuarios en la misma o en distintas multiconferencias, por lo que se decidió estudiar otras soluciones.

4.3.6.2 Hardware *Quicknet*

Otra de las opciones que se planteó, y también debido a la información que existe en la página web de *OpenH323 Project*, fue la de usar hardware específico.

Se informaba en la página web de las tarjetas *Quicknet* [14], que implementaban el codec G.723.1. Sin embargo una vez que se dispuso de este hardware se comprobó que estaba orientado a la posibilidad de añadir dicho codec a los programas clientes que ellos implementan (*openPhone* y *ohPhone*), no siendo posible unir, a priori, la tarjeta con la MCU.

Se intentó ver cómo podría ser esta integración pero, debido a la escasa información que acompañaba a este hardware, se descartó esta solución por el alto coste en tiempo que llevaría su implementación, si es que ésta fuera posible.

4.3.6.3 Cortocircuito en modo punto a punto

Básicamente la MCU debe hacer el siguiente tratamiento con el audio que le llega para cada sala: si en una habitación hay tres usuarios hablando, A, B y C, cuando la MCU debe enviarle el audio a C debe tomar el audio que le ha llegado de A y decodificarlo, el resultado deberá mezclarlo con el audio de B decodificado, y lo que obtiene deberá codificarlo (según el codec que haya negociado con C) antes de enviárselo.

La solución que al final se implementó fue establecer un cortocircuito en esta secuencia, sin embargo para que funcione sólo puede haber dos usuarios en la sala en la que se realiza el cortocircuito, A y B, el audio que llega de A se le envía sin ningún tratamiento a B, y viceversa. Además, para que ambos se entiendan deben estar usando el mismo codec.

Por tanto se decidió establecer dos escenarios básicos para las videoconferencias:

- **Videoconferencias punto a punto:** en una misma sala sólo dos usuarios, en nuestro caso un cliente y un asesor. Además estos usuarios deberán usar el mismo codec de audio tanto para transmitir como para recibir.

- **Videoconferencias multipunto:** se puede dar el caso de que existan más de dos usuarios en cada sala, y no tienen que utilizar el mismo codec. Se corresponde con el funcionamiento original de la MCU.

Estos dos escenarios se pueden dar simultáneamente en distintas salas sin afectar a la calidad de sonido en cualquiera de ellas.

Cortocircuito:

Después de estudiar la librería *OpenH323* se decidió implementar el cortocircuito en sí en el fichero *channels.cxx*, realizando una mezcla de las funciones `void H323_RTPChannel::Transmit()` y `void H323_RTPChannel::Receive()`. Necesitábamos ahora diferenciar cuándo había que realizar el cortocircuito y cuándo no.

Para hacer esta distinción se decidió que en modo punto a punto los usuarios deberían usar MS-GSM, y cualquier otro cuando trabajan en modo multipunto. De esta forma sólo se realizará el cortocircuito cuando se use el dicho codec, y esta distinción se implementa según:

```
if ((connection.GetSessionCodecNames
   (RTP_Session::DefaultAudioSessionID) == "MS-GSM"))
```

Por otro lado ahora se necesita saber quién está conectado en la misma sala que un determinado usuario, esto es, es necesario saber a quién enviar el audio cuando se realiza el cortocircuito. Dentro del fichero *channels.cxx* no se tiene acceso a las instancias de las clases *MyH323Connection* y *MyEndPoint*, definidas en la MCU, en las que se establecen estas relaciones, por lo cual se modificó la clase *H323Connection*, definida en el fichero *h323con.h* de la librería *OpenH323*. Se añade ahí la propiedad:

```
PString epConectado;
```

Que almacena el *token* que identifica con quién está conectado cada usuario. También se definen los métodos:

```
PString GetEpConectado();
void SetEpConectado(PString ep);
```

para su consulta y modificación.

Sin embargo esta variable se actualiza en el fichero *main.cxx* de la MCU, dentro de la función:

```
void MyH323EndPoint::AddMember(MyH323Connection * newMember)
```

en ella, cuando se conecta un usuario a una sala, se comprueba si ya hay alguien en esa sala, en cuyo caso se actualiza `epConectado` para las instancias de las conexiones de ambos usuarios.

Negociación del codec

Además se necesita que, en función del modo en el que un usuario va a trabajar (punto a punto o multipunto), la negociación de los codec que realiza con la MCU tenga como resultado el codec MS-GSM o no. La MCU necesita saber a priori en qué modo va a trabajar cada usuario para poder forzar la negociación, se decidió que esta información estuviera en la segunda línea del fichero de acceso de cada uno. Un 0 indica punto a punto, y un 1 multipunto.

Por tanto, ahora la MCU tiene que asegurarse que un usuario que se conecte en modo punto a punto negocie con la MCU sólo el codec MS-GSM, y cuando se conecte en modo multipunto tome cualquier otro codec.

OpenMCU tenía una sola tabla con los codec que era capaz de usar, en base a esta tabla se hacía la negociación del codec con cada usuario que se conectaba. Una vez que esta tabla se cargaba al principio ya no se modificaba, era la misma para todos los usuarios.

Para obligar que cuando un usuario concreto se conecte use un codec determinado (en nuestro caso MS-GSM), aunque éste pueda trabajar con algún otro que en principio le ofrezca más calidad, se tiene que gestionar en el momento de la negociación de los codec. El usuario negocia con la MCU un codec para transmitir y otro para recibir, no tienen que ser los mismos, y esto se hace en base a las tablas de codecs del usuario (tabla remota) y la tabla de codec de la MCU (tabla local). Para conseguir que en modo punto a punto se asegure que se negocie MS-GSM tanto para recibir como para transmitir se iba a tener que modificar una de

las dos tablas de codecs, de forma que sólo tuvieran este codec a la hora de negociar, en consecuencia sería el codec elegido tanto para recibir como para transmitir.

En un principio se pensó en modificar la tabla remota (del usuario), y no modificar la de la MCU, sin embargo no se podían eliminar codec del usuario desde la MCU, por tanto habría que modificar la tabla local. Había que asegurar que en el momento de la negociación de un usuario con perfil punto a punto la tabla local solo tuviera el codec MS-GSM, y si su perfil es multipunto la tabla tuviera todos los demás. Para no tener que estar añadiendo y quitando codecs continuamente de esta tabla, según se conectara un usuario con un perfil u otro, se decidió tener dos tablas de codec locales, una para punto a punto que sólo tuviera MS-GSM y otra para multipunto, con el resto de los codec que la MCU entiende.

Se definieron en el fichero *main.h* de la MCU, dentro de la clase *MyH323EndPoint* según:

```
H323Capabilities capabilitiesPTOaPTO;
H323Capabilities capabilitiesMULTIPUNTO;
```

Y también los métodos:

```
const H323Capabilities & GetCapabilitiesPTOaPTO() const
 { return capabilitiesPTOaPTO; }
const H323Capabilities & GetCapabilitiesMULTIPUNTO() const
 { return capabilitiesMULTIPUNTO; }
void RemoveCapabilitiesPTOaPTO(const PStringArray & codecNames)
 {capabilitiesPTOaPTO.Remove(codecNames);}
void SetCapabilitiesPTOaPTO (H323Capabilities cap)
 { capabilitiesPTOaPTO = cap; }
void SetCapabilitiesMULTIPUNTO (H323Capabilities cap)
 { capabilitiesMULTIPUNTO = cap; }
```

Cuando un usuario se conecta se le pasa una tabla u otra para la negociación, según se defina en su fichero de acceso. Esto se hace en la función:

```
H323Connection * MyH323EndPoint::CreateConnection
```

```
(unsigned callReference, PString nombreRemoto)
```

del fichero *main.cxx* de la MCU, por tanto en ella se tiene que identificar al usuario que intenta establecer la conexión para poder leer su fichero y pasarle una tabla u otra. Originariamente sólo recibía como parámetro `callreference`, por lo que era imposible identificar al usuario en este punto. Para que esta función reciba el último parámetro se tuvieron que modificar los ficheros que definen la clase *H323EndPoint*, de la librería *OpenH323*: *h323ep.h* y *h323ep.cxx*, haciendo que las funciones `callreference` allí definidas pasaran este parámetro.

Problema del vídeo

Sin embargo se siguió observando que cuando los usuarios trabajaban con módem, y se envíaba/recibía audio y vídeo, la calidad del sonido no era buena, por lo que se decidió tener la posibilidad de restringir el vídeo para determinados usuarios, ya que si no se trabaja con vídeo había una calidad de audio aceptable.

El código de *OpenMCU* tiene la posibilidad de indicar si se trabajaba o no con vídeo (opción `-v`), pero esto es aplicable a todos los usuario y todas las multiconferencias, lo que se implementó fue poder tomar esta decisión para cada usuario independientemente. Ahora, cuando se usa la opción se negociará el vídeo con cada usuario por separado. Si no se usa esta opción nadie trabajará con vídeo.

Hay que indicar a la MCU si un usuario tiene la posibilidad de enviar/recibir vídeo incluyendo en el fichero de acceso de cada usuario una nueva línea, de forma que este tendría la siguiente estructura: la primera línea sería el nombre de la habitación a la que iría cada usuario, la segunda línea sería un 0 (modo punto a punto) o un 1 (modo multipunto), y la tercera línea un 0 (no trabaja con vídeo) o un 1 (sí puede trabajar con vídeo).

Ambas tablas de codec (multipunto y punto a punto) contienen el codec de vídeo, pero justo antes de pasar la tabla al usuario para la negociación se comprueba en el fichero de acceso la capacidad de éste para trabajar con vídeo, si no puede, este codec se eliminará de la tabla que se le pase, de esta forma el usuario no podrá negociar el vídeo con la MCU y por tanto ésta no abrirá los canales lógicos de vídeo para ese usuario, lo que produce un mejor aprovechamiento del ancho de banda que deriva en una calidad de sonido mejor.

Sólo dos usuarios en habitación en modo punto a punto

Por último había que asegurar que en una sala con comunicación punto a punto no pudieran estar más de dos usuarios.

Se necesitaba comprobar si en la sala a la que iba a unirse alguien en modo punto a punto ya había más de una persona, en cuyo caso se le tendría que denegar el acceso. Esta comprobación se implementa en la función:

```
H323Connection::AnswerCallResponse
  MyH323Connection::OnAnswerCall(const PString & caller,
                                 const H323SignalPDU & setupPDU,
                                 H323SignalPDU & /*connectPDU*/)
```

Y se implementó la función:

```
Int MyH323EndPoint::NumUsuEnHabitacion(PString nombreHabitacion)
```

dentro de la clase *MyH323EndPoint*, que nos da el número de usuarios que hay en la sala `nombreHabitacion`.

4.4 Interfaz Gráfica

Para incluir una interfaz gráfica de usuario que nos permita la interacción con parte del software desarrollado de manera que se pueda consultar fácilmente los datos, se ha elegido la librería Qt, dado que reúne todas las características necesarias, y es fácil e intuitiva de utilizar.

Qt es una librería C++ que permite el desarrollo de interfaces gráficas multiplataforma. Qt está totalmente orientada a objetos, es fácilmente extensible, y proporciona verdadera programación de componentes mediante el mecanismo SIGNALS-SLOTS. Qt es la librería nativa de KDE, el entorno de escritorio de Linux.

Una de las principales características de Qt, es sin duda, su orientación multi-plataforma. De esta manera, podemos escribir aplicaciones portables a MS/Windows, Unix/X11, Mac y sistemas embebidos, con un mínimo de restricciones inherentes a las diferencias entre sistemas (nombres de ficheros, llamadas al sistema, concurrencia, ...). Qt es un producto de la compañía noruega *Trolltech* [9], y se distribuye en diferentes versiones, gratuitas (GPL) o comerciales. El modelo de objetos de C++ nos proporciona un soporte muy

eficiente, en tiempo de ejecución, del paradigma de POO. Pero además de la eficiencia, a la hora de desarrollar GUI's requerimos cierta flexibilidad y mecanismos de los que C++ carece.

Qt se basa en la herencia desde la clase *QObject*, utilizando técnicas estándar de C++, y su compilador de meta-objetos (MOC). Así, el modelo de objetos de Qt adquiere características de un modelo basado en componentes (por el bajo acoplamiento entre sus partes), mucho más capacitado para el desarrollo de GUI's. Para la creación de interfaces, QT incluye una potente herramienta de desarrollo visual, *QTdesigner*. Con esta herramienta, es posible crear la interfaz y programar su comportamiento, bien mediante código C++, bien mediante sus mecanismos visuales.

Sin duda, la principal característica de Qt, es su mecanismo de comunicación entre objetos SIGNALS-SLOTS. Este mecanismo, nos permite la interconexión de los eventos generados por un objeto (señales) a alguna funcionalidad de otro objeto (*slot*). Tanto las señales como los *slots* pueden ser los predefinidos en los objetos de Qt, o creados por el usuario. De esta manera, podemos generar una reacción simple o compleja de manera fácil.

Para poder compilar con las librerías Qt se ha modificado el fichero *Makefile*, para indicar la ruta de éstas, incluyendo también los nuevos ficheros.

En el diseño de la interfaz se ha valorado más la simplicidad y la funcionalidad que unos excesivos niveles de configuración, básicamente lo que se pretende es facilitar la consulta de los datos relativos a las conexiones en cada habitación.

Se ha incluido una nueva opción a la MCU, `--disable-interfaz`, la cual permite decidir si se mostrará o no la interfaz gráfica cuando se ejecute. El menú en modo texto seguirá funcionando simultáneamente a la interfaz gráfica, ésta se ejecuta en una hebra distinta.

Para el diseño de las pantallas se ha utilizado la herramienta *QtDesigner*, con la que se obtienen los ficheros con extensión `.ui`. Sin embargo, el código asociado se ha escrito en ficheros de texto independientes para poder realizar la compilación conjunta con la MCU y las librerías que ésta usa (*OpenH323* y *PWLib*).

La pantalla principal se muestra en la figura 4-2. En la parte superior hay tres botones, cuyas funciones son:

Refresca los datos que se muestran en la pestaña *Habitaciones*.

Muestra la siguiente ventana de información:

Sale del interfaz gráfico.

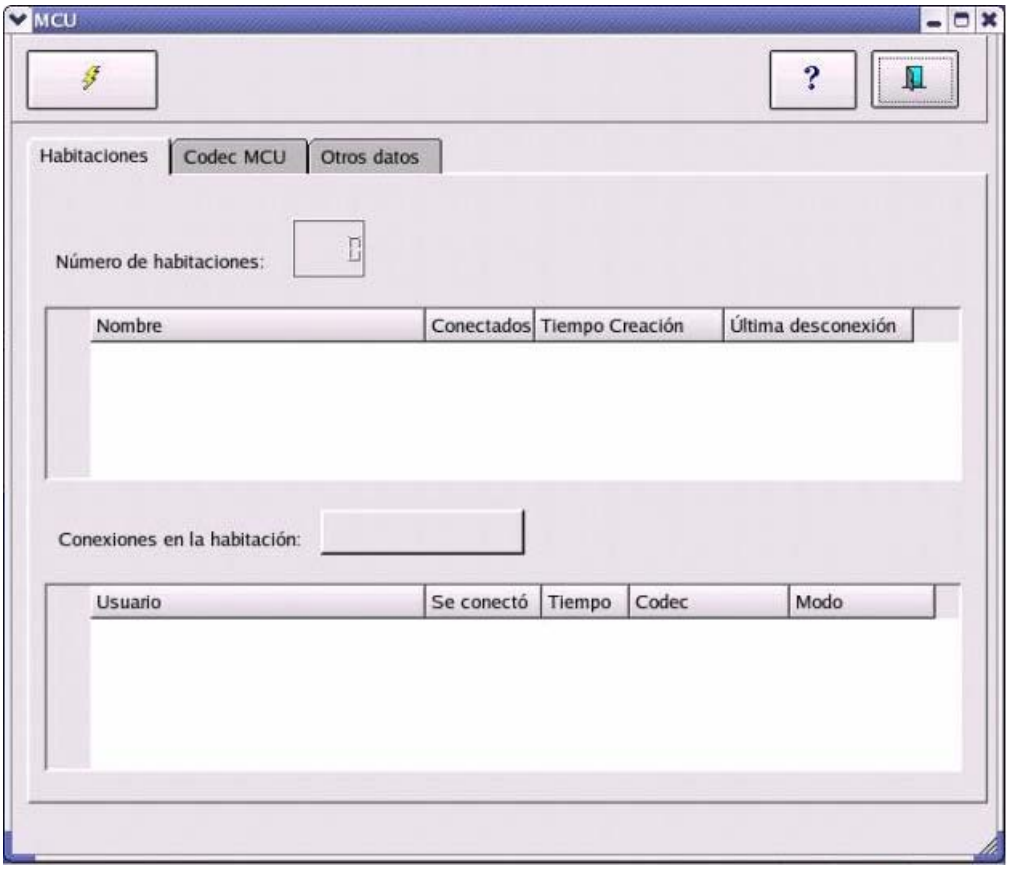

Figura 4-2. Pantalla de inicio

4.4.1 Pestaña Habitaciónes

Una vez que se pulsa el botón  se rellenan las tablas y los campos según se puede ver en la figura 4-3. Si no existe ninguna habitación aparece un mensaje en la parte inferior de la pantalla indicando esta circunstancia.

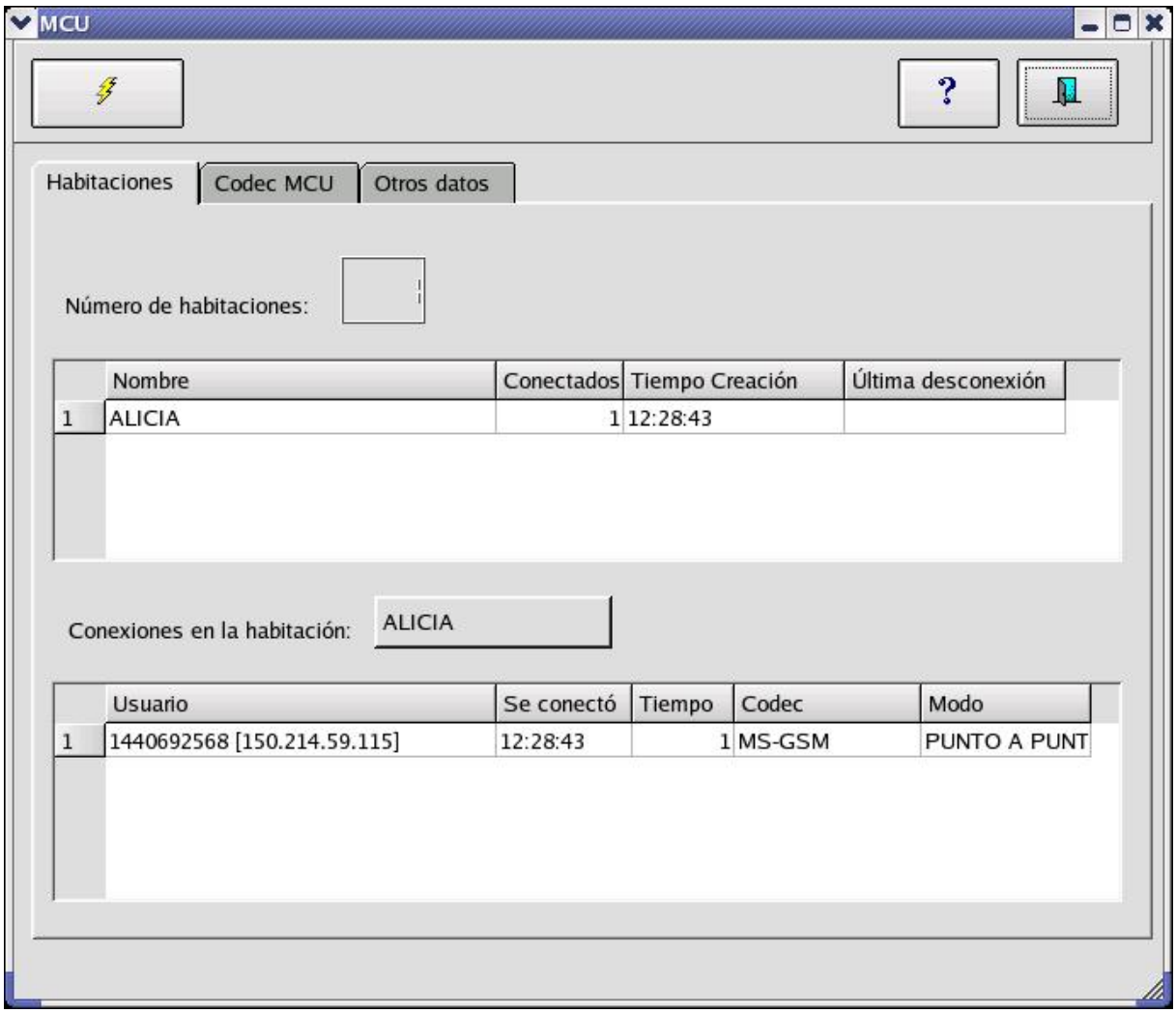

Figura 4-3. Pantalla inicio con datos

Como se puede ver en la figura, la primera tabla muestra información de las habitaciones creadas en el momento de la consulta:

- Nombre: es el nombre de la habitación.
- Conectados: indica el número de usuarios conectados en ese momento en dicha habitación.

- Tiempo Creación: es la hora en la que se creó la habitación.
- Última desconexión: muestra cuándo se desconectó algún usuario de la habitación, si es que alguno lo hizo.

Por otro lado, la tabla inferior muestra datos de las conexiones en cada habitación. En cada momento mostrará los datos de las conexiones relativas a la habitación cuyo nombre aparece en el campo que hay entre las dos tablas, que será la habitación seleccionada en la primera tabla. Para mostrar los datos de otra habitación simplemente la seleccionaremos en la primera tabla.

- Usuario: cadena que identifica al usuario conectado.
- Se conectó: hora en la que se conectó el usuario a la habitación.
- Tiempo: tiempo en segundos que lleva conectado.
- Codec: codecs de audio que utiliza.
- Modo: indica si trabaja en modo punto a punto o mulipunto.

4.4.2 Pestaña Codec MCU

Nos muestra las tablas de los codec con las que trabaja la MCU, tal y como se muestra en la figura 4-4.

Se muestran aquí las dos tablas de codecs locales:

- Tabla codecs punto a punto: es la tabla que se pasa a aquellos usuarios que realicen conexiones en modo punto a punto, según se indica en su fichero de acceso.
- Tabla codecs multipunto: se pasa a aquellos usuarios que trabajan en modo multipunto.

Ambas tablas contienen por defecto el codec de vídeo H.261-QCIF, sin embargo a la hora de pasarlas a los usuarios para la negociación puede eliminarse según se haya configurado la MCU o los ficheros de acceso.

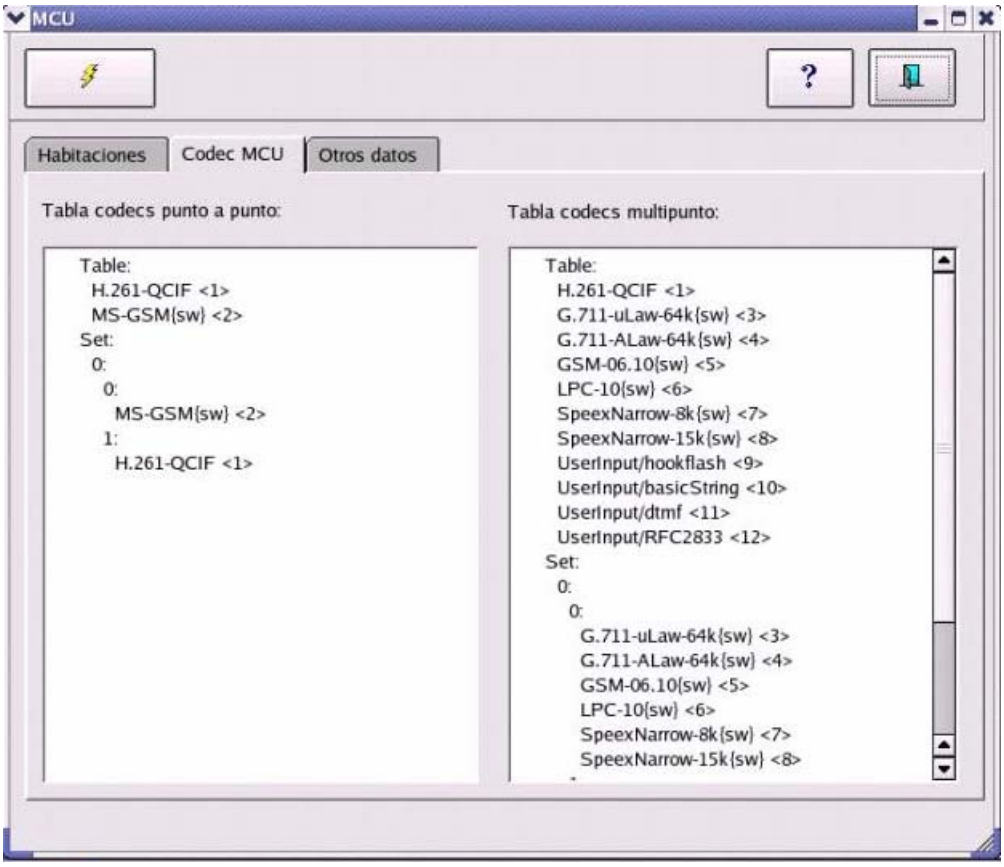

Figura 4-4. Pestaña codecs

4.4.3 Pestaña Otros Datos

Se muestran aquí otros datos que usa la MCU y se permite modificar el nombre con el que ésta será identificada por los usuarios que la llamen, esto es, el nombre que les aparecerá en *Netmeeting* como interlocutor.

Los datos que muestra son:

- Ruta de los ficheros: es la ruta en la que la MCU busca los ficheros de acceso y de eliminación, la que se pasa como parámetro cuando se ejecuta la MCU.

- *Jitter* mínimo y máximo: muestras los valores en los que se ha configurado el *jitter* para la ejecución de la MCU.

- Ancho de banda inicial: es el ancho de banda con el que inicialmente trabaja la MCU, aunque éste se verá limitado a los valores que realmente se puedan usar.

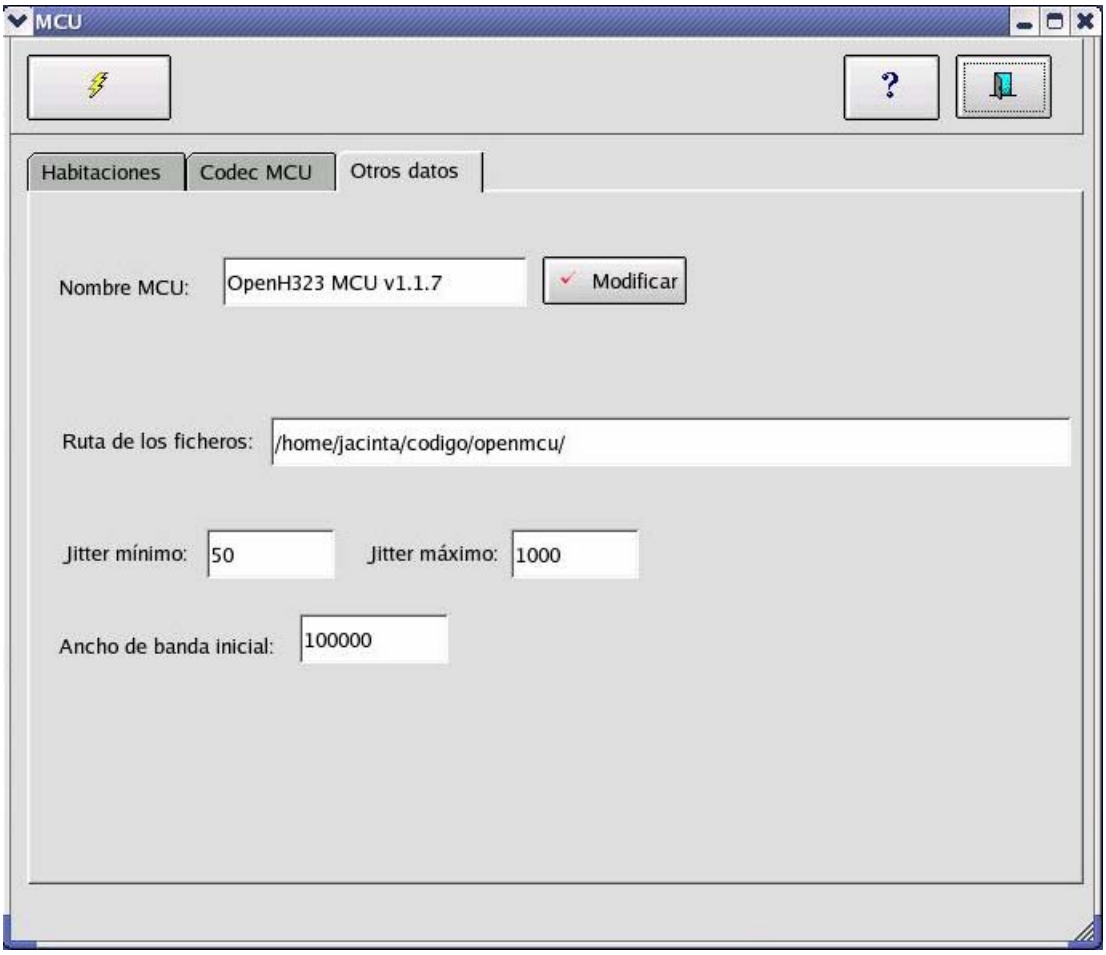

Figura 4-5. Pestaña Otros Datos

4.5 Pruebas

Para realizar las pruebas se han utilizado hasta 6 ordenadores conectados en red, éstos se describen en el apéndice C.

Al principio las pruebas se realizaron sólo para comprobar que las modificaciones realizadas funcionaban correctamente. Una vez que se vio que todo funcionaba adecuadamente se centraron en comprobar la calidad de audio y vídeo en las videoconferencias, para determinar si se permite una comunicación clara entre los usuarios, viendo así la viabilidad del producto. Se han realizado pruebas según los siguientes escenarios:

- 2 usuarios en la misma sala.
- 2 usuarios en distinta sala.
- 3 usuarios en la misma sala.

- 4 usuarios en 2 salas distintas.
- 6 usuarios en 3 habitaciones distintas.

Según estas posibilidades se fue cambiando el tipo de conexión, el ancho de banda y el perfil (asesor o cliente), el modo de conexión (punto a punto o multipunto), así como la posibilidad de trabajar o no con vídeo, de los usuarios que se conectaban, cambiando los ficheros de acceso de los mismos. También se modificaba el ancho de banda en *NetMeeting* para cambiar los codecs disponibles.

En todos los escenarios se ha obtenido una calidad de audio aceptable, y se puede trabajar con vídeo siempre que el usuario no se conecte a través de un módem, ya que el ancho de banda de que éste dispone no admite el audio y el vídeo simultáneo.

Dos de los ordenadores con los que se ha trabajado están bajo el sistema operativo Linux (uno en el que está instalada la MCU). Éstos no disponen de tarjetas de sonido ni cámaras web, sin embargo su uso como clientes está justificado para comprobar que no disminuye la calidad de sonido en el resto si se incrementa el número de habitaciones, así como para observar la correcta distribución de los usuarios en las distintas salas.

La mayoría de las pruebas se han realizado con el programa cliente *NetMeeting*, sin embargo los ordenadores con Linux usan el programa *ohphone*, de *OpenH323 Project*, y también se ha llegado a usar el programa *openPhone*, que su versión para Windows, comprobando que todo funcionaba correctamente independientemente del programa usado para la comunicación.

Capítulo 5: Conclusiones

Se ha modificado el código de *OpenMCU* de *OpenH323 Project* hasta adaptarlo a los requisitos del problema, realizando todas las modificaciones previstas en el anteproyecto, y algunas más, hasta obtener una MCU funcional que permite una comunicación clara entre usuarios distribuidos en distintas multiconferencias.

Las modificaciones realizadas han sido:

- Tomar las opciones de configuración directamente de un fichero. También se han añadido nuevas opciones (`–P, -D, --disable-intefaz`).
- Se ha establecido un control de los usuarios con los ficheros de acceso, permitiéndoles que se unan o no a una multiconferencia, uniéndolos a la habitación que se indique en su fichero, si existe.
- Se han implementado nuevos casos de desconexión de usuarios:
    - Cuando se excede el tiempo máximo de espera se desconecta al usuario y se elimina la habitación.
    - Cuando aparece un fichero de desconexión se desconecta al usuario concreto.
    - Cuando se va el asesor se desconectan todos los que estaban en su misma sala.
    - Cuando sólo quedan dos usuarios en una habitación y uno se va, entonces se desconecta también al otro usuario.
- Por requisitos en de diseño del proyecto CIMA, sólo se muestra el vídeo de los asesores.
- Se han almacenado tanto la hora de creación de cada habitación, como la hora a la que se desconecta de cada habitación el último usuario, si alguno lo hace, permitiendo consultar estos datos en cualquier momento incorporando al menú la opción “t”.

- Se han añadido otras opciones al menú, tales como “i” que nos muestra una lista de las conexiones y el modo de conexión de cada una. Y la opción “p” que nos lista todos los codecs remotos para cada conexión.

- Se han definido dos escenarios para las videoconferencias:

    - Videoconferencias punto a punto: en una misma sala sólo dos usuarios que deben usar el mismo codec de audio tanto para transmitir como para recibir.

    - Videoconferencias multipunto: más de dos usuarios en cada sala y no tienen por qué usar el mismo codec.

  De forma que se establece un cortocircuito en las videoconferencias punto a punto que nos permite una comunicación clara cuando los usuarios no disponen de ancho de banda suficiente. Para implementar este cortocircuito también se han tenido que definir dos tablas de codec locales para la MCU, en lugar de una como tenía originalmente, de forma que existe una tabla para videoconferencias punto a punto y otra para videoconferencias multipunto.

- Se ha permitido la negociación de trabajar o no con vídeo de forma individual con cada usuario según se indique en su fichero de acceso, aunque se puede restringir el uso del vídeo para todos los usuarios (con la opción `-v`).

Por otro lado también se ha desarrollado una interfaz gráfica que facilita la consulta de los usuarios conectados en cada momento a la MCU.

Se ha desarrollado una documentación que, aún no siendo robusta, si puede servir de base a futuros trabajos en este campo.

Se han realizado distintas pruebas al sistema para comprobar la calidad en las comunicaciones que la MCU proporciona, llegando a las siguientes conclusiones:

- En modo multipunto (el usuario se conecta a través de una red de área local) se obtiene una calidad de audio y vídeo óptima, independientemente de los usuarios conectados y de las habitaciones creadas.

- En modo punto a punto (el usuario se conecta a través de módem o ADSL) se obtiene una buena calidad de audio, siempre que no se mande vídeo, si se conecta a través de módem. Sin embargo se puede trabajar correctamente con audio y vídeo simultáneamente en otros casos (ADSL, RDSI).

por lo que las líneas futuras principales que puede tener este trabajo deberían estar encaminadas a la investigación, y desarrollo en su caso, de codecs de audio para ancho de banda bajo, que mejoren la calidad obtenida.