UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

Desarrollo de una aplicación de AudioConferencia sobre una plataforma ATM en entorno Linux

TESIS

Para optar el Grado de Maestro en Ciencias

Mención: Telemática

Presentada por:

César Núñez Ocola

LIMA - PERU

2000

UNIVERSIDAD NACIONAL DE INGENIERIA

Facultad de Ingeniería Eléctrica y Electrónica

Desarrollo de una Aplicación de AudioConferencia sobre una Plataforma ATM en Entorno Linux

TESIS

Para optar el grado de Maestro en Ciencias

Mención: Ingeniería Electrónica. Especialidad: Telemática

Presentada por:

César Núñez Ocola

Lima - Perú

Extracto

Durante mucho tiempo , se ha orientando la aplicación de ATM en el desarrollo de ruteadores y conmutadores empleados en las "columnas vertebrales"de redes de alta velocidad de fibra óptica. Pero recientemente , la tecnología ATM ha empezado ha jugar un rol importante en la evolución de las redes de trabajo en grupo, de campus y de empresas. Hoy en día es posible conformar verdaderas redes LAN ATM de terminal a terminal, gracias a la existencia de Tarjetas de Interface ATM que permiten interconexión entre computadores. Con el fin explotar la tecnología ATM llevada al campo de los terminales finales, los diversos sistemas operativos han dado inicio a la creación de manejadores e interfaces de programación de aplicaciones (API) ATM en base a los cuales es posible el desarrollo de aplicaciones tanto nativas ATM, como de integración de otros estándares sobre dicha tecnología.

Sin embargo el desarrollo de aplicaciones ATM nativas, basadas en el empleo de un determinado API, acarrea tanto ventajas como problemas de diseño, debidos tanto al sistema operativo como a la naturaleza de las especificaciones ATM en base a los cuales esta dada su arquitectura.

El trabajo de la presente tesis se basa en en el estudio del API ATM desarrollada sobre el sistema operativo Linux, la cual a pesar que su desarrollo está aún en marcha, provee hoy en día la mayoría de las funcionalidades esperadas en el "estado

de arte"de las redes LAN ATM, las mismas que se basan en las especificaciones del Forum ATM para transporte de datos de red.

En el trabajo de la presente tesis se desarrolla una plataforma de red ATM en base a la arquitectura de la API ATM del sistema operativo Linux, sobre la cual se plantea el desarrollo de una aplicación de AudioConferencia, la misma que engloba los siguientes objetivos específicos:

1. Establecer un esquema de manejo de difusión sobre la plataforma de red ATM implementada.

2. Evaluar las facilidades ofrecidas por el sistema operativo Linux para el desarrollo de aplicaciones adquisición, tratamiento y transmisión de información de audio en tiempo real.

3. Evaluar la performance de la especificación AAL5 de la plataforma red ATM implementada para el transporte de información de audio digitalizado de alta calidad.

Tabla de Contenido

Capítulo 1

Introducción

El Modo de Transferencia Asíncrono (ATM) es una tecnología de red en constante evolución diseñada para transportar flujos de tráfico heterogeneo que incluye video, voz y datos. ATM provee un servicio orientado a la conexión con entrega del mejor esfuerzo, pudiendo proveer una calidad de servicio garantizada.

Una conexión ATM esta dada entre dos puntos terminales de una red y es llamada Conexión de Canal Virtual. Dos tipos distintos de conexiones son soportados. Una conexión de Canal Virtual Permanente (PVC) es análogo a una linea rentada en una red telefónica. Este debe ser manualmente configurado por un administrador de sistema y persiste hasta que el administrador lo destruye manualmente. Un Canal Virtual Commutado (SVC) es análogo a una conexión por llamada en la red telefónica. Este es creado en respuesta a una petición por parte de uno de los puntos finales y persiste únicamente hasta que uno de los puntos finales termina la conexión.

La célula ATM de 53 bytes es la unidad básica de transporte de datos en una red ATM. Para adaptar los mecanismos de transporte de células ATM a los requerimientos específicos de las distintas aplicaciones tanto en tiempo real y tiempo no real, se han definido algunos protocolos de transporte comunmente llamados capas de adaptación ATM (AALs). Estos protocolos de adaptación estan presentes únicamente en los nodos finales de una red ATM. El más significativo de estos AALs es AAL5, el cual, es el estándar defacto para todo tráfico con Taza de Bit Variable (VBR) en redes de computadores basadas en ATM, conocidas como LAN ATM.

La aplicación de la tecnología ATM en el campo de redes de computadores o estaciones finales, ha generado en la industria de la computación el desarrollo de nuevos elementos tanto a nivel de hardware (tarjetas interface ATM) como de software (desarrollo de drivers y APIs ATM) a fin de poder explotar las bondades ofrecidas por esta tecnología. Un ejemplo de esto, es el desarrollo del soporte de protocolo ATM en Linux, el cual viene siendo coordinado por Werner Almesberger (el autor de LILO) en el LRC (Laboratoire de Reseaux de Communication) en Lusanne, Suiza.

1.1 Formulación del problema

La eficiencia de la tecnología ATM para el transporte de información multimedia entre terminales ATM, ha sido probada y demostrada en diversos estudios realizados, sin embargo el empleo de dicha tecnología en el desarrollo de determinadas aplicaciones acarrea problemas de diseño debido a su naturaleza de operación, conexiones punto a punto a nivel de terminales ATM, uno de estos casos es el desarrollo de aplicaciones basadas en multidifusión, como aplicaciones de AudioConferencia donde la información de audio es difundida a todos los participantes de una sesión determinada, por medio del manejo de conexiones punto - multipunto.

Una solución clásica a este problema es contar con un switch ATM capaz de manejar conexiones punto - multipunto, a partir del cual se realiza la respectiva réplica y difusión de células ATM, a todos y cada uno de los participantes de una sesión de AudioConferencia, exceptuando lógicamente el participante origen de la información.

La hipótesis de la presente tesis se fundamenta en el empleo de terminales finales ATM organizados en un esquema de arbitraje centralizado, interconectados por medio de conexiones virtuales permanentes AAL5, con especificación de calidad de servicio UBR para el transporte de información audio de alta calidad digitalizado.

1.2 Objetivos de la presente tesis

1. Desarrollo de una plataforma de red ATM en base a la arquitectura del API ATM sobre el sistema operativo Linux.

2. Establecer un esquema de manejo de difusión sobre la plataforma de red ATM implementada.

3. Evaluar las facilidades ofrecidas por el sistema operativo Linux para el desarrollo de aplicaciones de adquisición, tratamiento y transmisión de información de audio en tiempo real.

4. Evaluar la performance de la especificación AAL5 de la plataforma red ATM implementada para el transporte de información de audio digitalizado de alta calidad.

1.3 Organización de la tesis

El Capítulo 2 presenta los conceptos y definiciones base de la tecnología ATM sobre la cual esta fundamentada la presente tesis, en el capítulo 3 se da una revisión de la plataforma Linux ATM, analizando sus alcances y limitaciones, los dos últimos capítulos están enfocados en el estudio de los elementos necesarios adicionales y el desarrollo de la aplicación de AudioConferencia.

Capítulo 2

Tecnología ATM

2.1 Modelo de referencia B-ISDN ATM

En los estándares de la ITU-T, se describe la B-ISDN como una evolución de la ISDN, basada en extensiones de la misma; brindando rangos para servicios de audio, datos, vídeo y multimedia en banda angosta y banda ancha. Los servicios pueden ser distribuidos o interactivos, con tasas de bit constantes (CBR) o variables (VBR), orientados a conexión o sin conexión. A manera de apoyo de estos servicios, la red brinda conexiones virtuales que pueden ser punto a punto o multipunto, simétricas o asimétricas, unidireccionales o bidireccionales y conmutadas o semipermanentes [6].

En el Artículo I.320 de la ITU-T se describe el PRM (modelo del protocolo de referencia) que se usa para ISDN. La importancia del PRM reside en que ha aportado el concepto de separación de funciones de usuario, control y manejo. El Artículo I.320 de la ITU-T constituye el fundamento para el PRM utilizado para B-ISDN; comprendido en el artículo I.321. Este último comprende la fisiología de la B-ISDN.

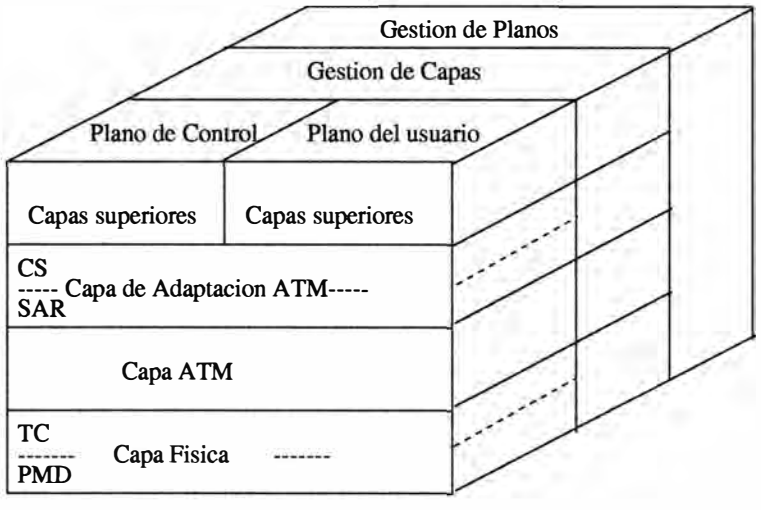

Figura 2.1: Modelo de referencia ATM-BISDN

El PRM B-ISDN consiste en tres planos [6]:

- Plano de Usuario
- Plano de Control
- Plano de Manejo

El Plano de Manejo está dividido en dos secciones, de acuerdo a las funciones que ejecutan: las funciones del gestor de capas y las funciones del gestor de planos. Todas las funciones de manejo relacionadas con todo el sistema se localizan en el gestor de planos, el cual es responsable de la coordinación entre todos los planos. En este plano no existe una división por capas.

El plano de Gestion de Capas está compuesto por una estructura de estratos. Lleva a cabo las funciones de manejo relacionadas con recursos y entidades que se encuentran en su protocolo.

El Plano de Usuario sirve para la transferencia de información del usuario. Incluye a todos los mecanismos asociados, como el control de flujo y recuperación de errores. Este plano representa los protocolos de acarreo de información entre los usuarios a través de la red. Su constitución interna consta de 4 niveles:

- Capas Superiores
- Capa de Adaptación de ATM (AAL)
- Capa ATM
- Capa Física

El funcionamiento de las capas es muy similar al del modelo de referencia ISO/OSI; utilizan los servicios que les brindan las capas inferiores y proveen servicios a las superiores. La estructura por medio de capas permite independencia en el diseño e implementación de cada capa.

Los servicios que se brindan al usuario se encuentran en la última capa de este plano. Con propósitos didácticos, la ITU-T ha clasificado los servicios en cuatro áreas principales. La Clase A comprende servicios orientados a conexión y con tasa de bit constante, como por ejemplo emulación de circuitos o discursos CBR. Las Clases B a D son servicios de tasa de bit variable. La Clase B contiene aquellos servicios que son orientados a la conexión con una relación de tiempo requerida entre la fuente y el destino; por ejemplo, vídeo VBR. La Clase C se diferencia de la B por que no requiere del compromiso de relación de tiempo; como por ejemplo X.25.

La Clase D hace referencia a los servicios sin conexión; por ejemplo, la transferencia de datos entre LAN's y MAN's.

La Capa de Adaptación de ATM (AAL) maneja todos los servicios arriba de la Capa ATM de servicio independiente. Las funciones de la AAL son de servicios específicos, y originalmente fueron diseñados con la numeración del uno al cuatro; en correspondencia a las clases de servicio A a D. Sin embargo, los modelos AAL3 y AAL4 se han unido para dar lugar a un solo modelo llamado AAL3/4 , y un nuevo modelo denominado AAL5 se ha adoptado a manera de una versión más simple del modelo AAL3/4 para servicios de datos que emplean VBR [1].

El AAL se compone de dos niveles; el Subnivel de Convergencia (CS) y el subnivel de Segmentación y Reensamblado (SAR). EL CS es una especificación de interface de servicio dependiente, la cual incluye multiplexión, control de error, detección de pérdida de celdas, y recuperación de los periodos de tiempo. El subnivel SAR divide la información de longitud variable proveniente del CS en celdas ATM para la Capa ATM y reconstruye las celdas ATM en sus unidades originales de CS. Las funciones de los modelos de AAL se llevan a cabo únicamente en los márgenes de la red, y la información del AAL se desplaza transparentemente dentro de una celda ATM (campo de información) en la Capa ATM.

La Capa ATM acepta unidades del AAL listas para la encapsulación (generación y concatenación de cabecera) y entrega la información al AAL después de la desencapsulación (extracción y procesamiento de cabecera) de la celda. La función principal de la Capa ATM es la transferencia secuencial de principio a fin de celdas ATM de acuerdo a la información de protocolo contenida en la cabecera de cada celda.

Por debajo de la Capa ATM, se encuentra la Capa Física, la cual es la responsable de la transmisión de las celdas ATM a manera de corrientes de bits a través del medio físico. Esta capa se divide en el subnivel de Medio Físico (PM) y subnivel de Convergencia de Transmisión (TC). El subnivel PM transmite una corriente de bits con información de tiempo asociada y codificación de línea, la cual depende de el medio físico. El subnivel TC se constituye por funciones de transmisión independientes del medio físico, entre las cuales se encuentran:

- Transmisión, generación de trama y recuperación.
- Adaptación de trama (mapeo de celdas dentro y fuera de la carga de la trama).
- Delineación de celdas (identificación y recuperación de los límites de las celdas).

- Procesamiento y generación de revisión de error de cabecera de celda.
- Ajuste de la velocidad de celdas (Inserción y extracción de celdas vacías).

La arquitectura de capas también es utilizada dentro del Plano de Control. Este plano es responsable de las funciones de control de llamadas y conexiones. Todas estas son funciones de señalización empleadas para establecer, supervisar y liberar una llamada o conexión. Las funciones de la Capa Física y de la Capa ATM (ATM Layer) son las mismas para el Plano de Control y el Plano de Usuario.

A continuación realizaremos un análisis capa por capa del modelo de referencia BISDN-ATM, siguiendo un esquema de arriba hacia abajo.

2.2 Capa de Adaptación ATM

Esta capa se encuentra entre la Capa ATM y las capas superiores. Su función básica consiste en aumentar los servicios de adaptación proporcionados por la Capa ATM, de acuerdo a los requerimientos de la capa superior. Las Unidades de Protocolos de Datos (PDU) de las capas superiores se mapean dentro del campo de información de la celda ATM. Esta capa proporciona el mapeo de los PDU's de las capas superiores dentro del campo de información de las celdas y rensambla los PDU's.

La Capa de Adaptación AAL, se divide en dos partes; una subcapa llamada de Segmentación y Ensamblaje (SAR) y la Subcapa de Convergencia (CS). Las funciones principales de la subcapa SAR son: en el transmisor, la segmentación de las PDU's de las capas superiores a fin de que tengan un tamaño que encaje en el campo de información de la celda ATM (48 bytes); por el lado del receptor, la función que debe cumplir es el ensamblaje de los campos de información particulares dentro de las PDU's de las capas superiores. El CS consiste en un servicio dependiente y proporciona el servicio AAL en el AAL-SAP.

No se ha definido aún ningún SAP entre estas dos subcapas. Los SAP para capas superiores se pueden derivar de las diferentes combinaciones de SAR y CS. Para algunas aplicaciones, no son necesarias ninguna de las dos subcapas ni el SAR ni el CS, en tal caso estarán vacíos.

Se han definido varios tipos de protocolos AAL; cada uno con subcapas SAR y CS específicas. Sin embargo no existe una relación estrecha entre los servicios AAL y los protocolos AAL. Pueden existir otras combinaciones de SAR y CS, o aún otros SAR y/o CS pueden utilizarse como respaldo de servicios específicos.

Para reducir el número de protocolos AAL, la ITU-T propuso una clasificación de servicios, con respecto a los siguientes parámetros: relación de tiempos, tasa de bits, y modos de conexión. En la tabla 2.2 se muestra un esquema de dicha clasificación. No todas las posibles combinaciones tienen sentido, por lo cual hay únicamente cuatro clases que se han distinguido [4]. Los servicios que ofrecen las cuatro clasificaciones se muestran en la figura.

Clase	A		B	C			D	
Sincronizacion	Tiempo real	Ninguna	Tiempo real	Ninguna	Tiempo real	Ninguna	Tiempo real	Ninguna
Tasa de bits	Constante		Variable		Constante		Variable	
Modo	Orientado a conexiones				Sin conexiones			

Figura 2.2: Clases originales de servicio reconocidas por la AAL

2.2.1 AAL tipo 1

AAL1 es el protocolo usado para transmitir tráfico clase A, es decir, tráfico orientado a conexiones de tiempo real y con tasa de bits constante, como audio o vídeo sin compresión. Los bits son alimentados por la aplicación a una velocidad constante, y deben entregarse en el otro lado a la misma velocidad constante, con retardo, fluctuación y carga extra mínimos. La entrada es una corriente de bits, sin límites de mensaje. Para este tráfico no se usan los protocolos de detección de errores como el de parada y espera, porque los retardos que generan las terminaciones de temporización y las retransmisiones no son aceptables. Sin embargo, las células faltantes se informan a la aplicación que entonces, si lo desea, puede tomar sus propias medidas para recuperarlas.

AAL1 tiene una subcapa de convergencia y una subcapa SAR. La primera detecta células perdidas y mal introducidas (Una célula mal introducida es aquella que se entrega al destino equivocado como resultado de un error no detectado en el identificador de su circuito virtual o trayectoria virtual). Esta subcapa también amortigua el tráfico de entrada para proporcionar entrega de células a una tasa constante. Por último, la Subcapa de Convergencia divide los mensajes o la corriente de entrada en unidades de 46 o 47 bytes que se entregan a la subcapa SAR para su transmisión. En el otro extremo se extraen estas unidades y se reconstruye la entrada original. La Subcapa de Convergencia de AAL1 no tiene ninguna cabecera

de protocolo propio.

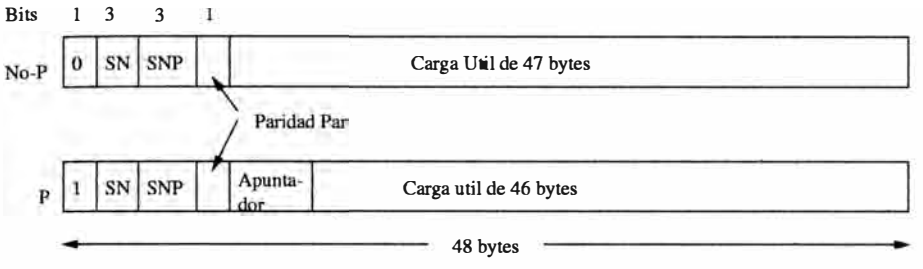

Figura 2.3: Formato de célula AAL1

En contraste, la subcapa SAR de AAL1 si tiene un protocolo. Los formatos de sus células se muestran en la figura 2.3. Ambos formatos comienzan con una cabecera de un 1 byte que contiene un número de secuencia de célula de 3 bits, SN, para detectar células perdidas o mal introducidas. A este campo le sigue un número de protección de secuencia (es decir, suma de comprobación) de tres bits, el SNP, basado en el número de secuencia, para permitir la corrección de errores individuales y la detección de errores dobles en el campo de secuencia. Esta suma usa una comprobación de redundancia cíclica con el polinomio $x^3 + x + 1$. Un bit de paridad par que cubre el byte de cabecera reduce aún mas la posibilidad de un número de secuencia equivocado.

Las células AAL1 no necesitan llevar 47 bytes . Por ejemplo, para transmitir voz digitalizada que llega a razón de 1 byte cada 125 useg, el llenado de una célula con 47 bytes implicaría la recolección de muestras durante 5.875 mseg. Si este retardo antes de la transmisión es inaceptable, pueden enviarse células parciales. En este caso, el número de bytes de datos reales por célula es igual para todas las células y se acuerda por adelantado.

Las células P se usan cuando deben preservarse los límites de los mensajes. El campo de apuntador sirve para indicar el desfasamiento del comienzo del siguiente mensaje. Sólo las células con un número de secuencia par pueden ser células P, por lo que el apuntador está en el intervalo de 0 a 92, para que apunte dentro de la carga útil de su propia célula o de la que sigue. Nótese que este esquema permite que los mensajes tengan una cantidad arbitraria de bytes por lo que pueden enviarse mensajes continuamente y no necesitan alinearse con los límites de la célula.

El bit de orden mayor del campo apuntador se reserva para uso futuro. El bit inicial de cabecera de todas las células de numero impar forman una corriente de datos usada para la sincronización de reloj.

2.2.2 AAL tipo 2

El AAL1 se diseño para corrientes de datos sencillas en tiempo real, orientadas a conexiones y sin detección de errores, excepto por células faltantes o mal introducidas. Para el audio o vídeo puro sin compresión, o cualquier otra corriente de datos en la que unos cuantos bits alterados de vez en cuando no representan un problema, es adecuado el AAL1.

En el audio o vídeo comprimido la tasa puede variar considerablemente con el tiempo. Por ejemplo, muchos esquemas de compresión transmiten un marco completo de vídeo, luego envían durante varios marcos solo las diferencias entre los marco subsiguientes y el último marco completo. Cuando la cámara es estacionaria, nada se mueve, los marcos de diferencia son pequeños. Pero cuando la cámara esta haciendo una panorámica rápida, son grandes. Además los límites de los mensajes deben conservarse para que pueda reconocerse el comienzo del siguiente marco completo aún en presencia de células perdidas o datos erroneos. Por estas razones se requiere un protocolo más complejo. Se diseño para estos fines el AAL2.

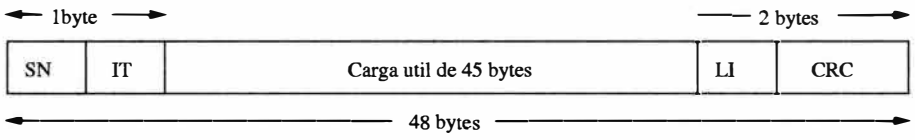

Figura 2.4: Formato de célula AAL2

Como el AAL1, la subcapa CS no tiene un protocolo pero la subcapa SAR sí. EL formato de célula SAR se muestra en la figura 2.4; tiene una cabecera de un byte y un apéndice de 2 bytes dejando espacio para 45 bytes de datos por célula.

El campo SN (sequence Number, número de secuencia) sirve para numerar las células en orden a fin de detectar células faltantes o mal introducidas. El campo IT (Information Type, tipo de información) indica que la célula es el comienzo, la mitad o el fin de un mensaje. El campo LI (Lenght Indicator, indicador de longitud) indica el tamaño de la carga en bytes (puede ser menor que 45 bytes). Por último el campo CRC es una suma de comprobación de la célula completa que permite detectar errores. Aunque pueda parecer extraño los tamaños de campo no se incluyen en el estándar. De acuerdo con un enterado, ya al fnal de proceso de estandarización, el comité se dió cuenta de que el AAL2 tenía tantos problemas que no debería usarse, desafortunamente era demasiado tarde para detener el proceso de estandarización, y había una fecha que cumplir. En un último esfuerzo el comité quitó todos los

tamaños de campo para que pudiera publicarse el estándar formal a tiempo, pero de manera tal que nadie pudiera usarlo.

2.2.3 AAL tipo 3/4

Originalmente, la ITU tenía protocolos diferentes para las clases C y D servicio orientado a conexiones y servicio sin conexiones para transporte de datos sensibles a pérdidas y errores, pero no depedientes del tiempo. Luego la ITU descubrió que no había necesidad real de dos protocolos por lo que los combinó en uno sólo, el AAL3/4.

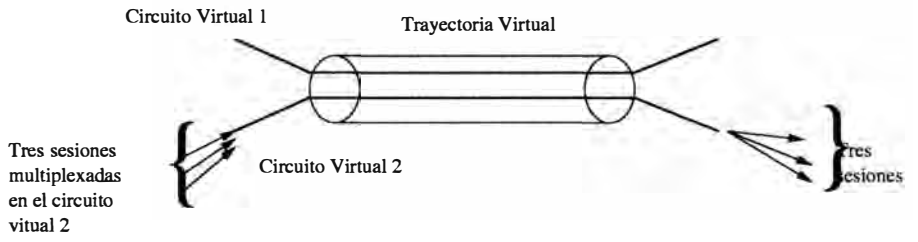

Figura 2.5: Multiplexión de varias sesiones en un circuito virtual

El AAL3/4 puede operar de dos maneras : corriente y mensajes. En el modo de mensajes cada llamada de la aplicación al AAL3/4 inyecta un mensaje en la red. El mensaje se entrega como tal, es decir , se conservan los límites del mensaje. En el modo de corrientes no se conservan los límites. El análisis siguiente se concentrará en el modo de mensajes. En cada modo hay disponibles transporte confiable y no confiable (es decir sin garantía).

Una característica del AAL3/4 no presente en los otros protocolos es la multiplexión, este aspecto del AAL3/4 permite que viajen por el mismo circuito vitual múltiples sesiones (por ejemplo, sesiones interactivas remotas) de un solo terminal final, y que se separen en el destino, como se ilustra en la figura 2.5.

La razón por la que es deseable esta facultad es que las portadoras con frecuencia cobran por cada establecimiento de conexión y por cada segundo que permanece abierta la conexión. Si un par de terminales tiene abiertas varias sesiones simultáneas, dar a cada una su propio circuito virtual será más caro que la multiplexión de todas ellas en el mismo circuito virtual. Si un circuito virtual tiene suficiente ancho de banda para manejar todo no hay necesidad de más de uno.

Todas las sesiones que usan un solo circuito virtual reciben la misma calidad de servicio ya que ésta se negocia para cada circuito virtual.

Esta cuestión es la verdadera razón por la que originalmente había formatos AAL3 y AAL4 separados: los estaunidenses querían multiplexión y los europeos nó, así que cada grupo creó su propio estándar, después, decidieron que ahorrar diez bits de cabecera no valían el precio de que Europa y Estados Unidos no se pudieran comunicar. Por el mismo costo, se habrían podido empecinar en sus puntos de vista y tendríamos cuatro estándares AAL incompatibles (de los cuales uno está incompleto) en lugar de tres.

A diferencia del AAL1 y el AAL2, el AAL3/4 tiene tanto un protocolo de Subcapa de Convergencia como uno de subcapa SAR. Los mensajes de hasta 65 535 bytes entran en la Subcapa de Convergencia desde la aplicación; primero se rellenan a un múltiplo de 4 bytes y luego se les agrega una cabecera y un apéndice, como se muestra en la figura 2.6.

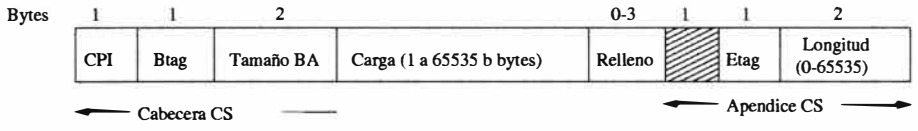

Figura 2.6: Formato de mensaje de Subcapa de Convergencia AAL3/4

El campo CPI (Common Part Indicator, indicador de parte común) indica el tipo de mensaje y la unidad de conteo de los campos tamaño BA y longitud. Los campos BTAG, y ETAG sirven para enmarcar los mensajes. Los dos bytes deben ser iguales y se incrementan en uno en cada mensaje nuevo enviado. Este mecanismo verifica si hay pérdida o mala introducción de células.

El campo tamaño BA se usa para asignación de buffers; indica al receptor la cantidad de espacio de buffer a asignar para el mensaje antes de su llegada. El campo de longitud da nuevamente la longitud de la carga útil; en el modo de mensaje debe ser igual al tamaño BA, pero en modo de corriente puede ser diferente. El apéndice también contiene un byte no usado.

Una vez que la Subcapa de Convergencia a destruído el mensaje y le ha agregado una cabecera y una cola, como se muestra en la figura 2.6, pasa el mensaje a la subcapa SAR, que lo divide en bloques de 44 bytes. Nótese que, para manejar multiplexión, la Subcapa de Convergencia, puede construír varios mensajes internamente al mismo tiempo y pasar bloques de 44 bytes a la subcapa SAR, primero de un mensaje luego de otro, en cualquier orden.

La subcapa SAR inserta cada bloque de 44 bytes en la carga últil de una célula cuyo formato se muestra en la figura 2.7. Estas células se transmiten entonces al

destino para su reensamble, tras lo cual se lleva a cabo la verificación de la suma de comprobación y se realiza alguna acción de ser necesaria.

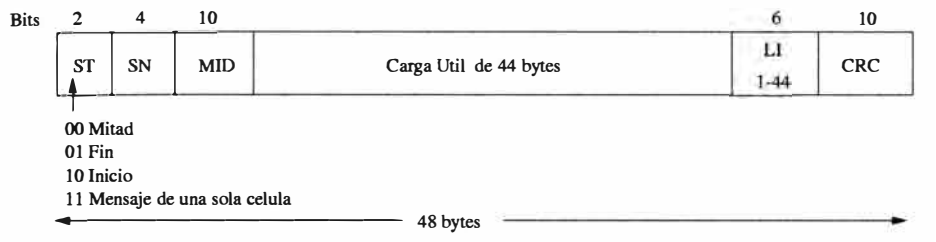

Figura 2.7: Formato de célula AAL3/4

Los campos de la célula AAL3/4 son los siguientes. El campo ST (Segment Type, tipo de segmento) sirve para enmarcar los mensajes; indica si la célula inicia un mensaje, está a la mitad de él, es la última célula o si es un mensaje pequeño (es decir, de una sóla célula). Sigue un número de secuencia de 4 bits, SN, para detectar células faltantes o mal introducidas. El campo MID (Multiplexing ID, identificador de multiplexión) sirve para llevar el registro de la sesión a la que pertenece cada célula. Recuerde que la Subcapa de Convergencia puede tener varios mensajes, pertenecientes a diferentes sesiones, en buffer al mismo tiempo y que puede enviar partes de estos mensajes en cualquier orden que desee. Todas las partes de los mensajes que pertenecen a la sesión I llevan I en el campo MID, para poderlos reagrupar correctamente en el destino. El apéndice contiene la longitud de carga útil y la suma de comprobación de la célula.

Nótese que el AAL3/4 tiene carga extra de dos capas de protocolo: 8 bytes que se agregan a cada mensaje y 4 bytes que se agregan a cada célula. En suma, es un mecanismo de peso pesado, especialmente para los mensajes cortos.

2.2.4 AAL tipo 5

Los protocolos AAL1 a AAL3/4 se diseñaron en gran medida para la industria de las telecomunicaciones y fueron estandarizados por la ITU sin mucha realimentación de la industria de la computación. Cuando la industria de la computación finalmente despertó y comenzó a entender las implicaciones de la figura 2.7, surgió una especie de pánico. La complejidad y la ineficiencia generada por dos capas de protocolo aunadas a la suma de comprobación extrañamente corta (sólo 10 bits) hizo que algunos investigadores inventaran un protocolo de adaptación nuevo. Lo llamaron SEAL (Simple Eficient Adaptation Layer, capa de adaptación sencilla y eficiente)

lo que sugiere qué pensaban estos diseñadores de los anteriores. Tras cierto debate, el Forum ATM aceptó el SEAL y le asignaron el nombre AAL5. Para mayor información sobre el AAL5 y sus diferencias respecto al AAL3/4 véase [9]

El AAL5 ofrece varios tipos de servicio a sus aplicaciones. Una posibilidad es el servicio confiable (es decir, entrega garantizada con control de flujo para evitar desbordamientos) otra posibilidad es el servicio no confiable (es decir, sin entrega garantizada), con opción de descartar o pasar a la aplicación (reportando como malas) las células con errores de suma de comprobación. Se apoyan tanto la unitransmisión como la multitransmisión, pero la multitransmisión no ofrece entrega garantizada.

Al igual que el AAL3/4, el AAL5 reconoce tanto modo mensajes como modo corriente. En el modo mensajes una aplicación puede pasar un datagrama de 1 a 65535 bytes de longitud a la capa AAL para la entrega a su destino ya sea con garantía o con el mejor esfuerzo. A su llegada a la Subcapa de Convergencia se rellena un mensaje y se le agrega un apéndice, como se muestra en la figura 2.8. La cantidad de relleno (0 a 47 bytes) se selecciona para hacer que el mensaje completo, incluído el relleno y el apéndice sea un múltiplo de 48 bytes. El AAL5 no tiene una cabecera de Subcapa de Convergencia sólo un apéndice de 8 bytes.

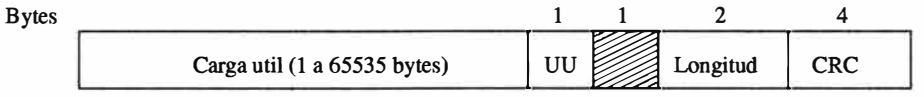

Figura 2.8: Formato de mensaje de Subcapa de Convergencia AAL5

El campo UU (User to User, usuario a usuario) no lo usa la capa AAL misma, una capa superior puede usarlo para sus propios fines; por ejemplo, números de secuencia o multiplexión.

La capa superior en cuestión puede ser la subparte específica para servicio de la subcapa de convergencia. El campo de longitud indica el tamaño de la carga útil real, en bytes, sin contar el relleno. Se usa un valor de 0 para abortar el mensaje actual a medio camino. El campo CRC es la suma de comprobación estándar de 32 bits del mensaje completo, incluído el relleno y el apéndice (con el campo CRC en 0). Se reserva un campo de 8 bits del apéndice para uso futuro.

El mensaje se transmite pasándolo a la subcapa SAR, que no agrega cabeceras ni apéndices; en cambio, divide el mensaje en unidades de 48 bytes y pasa cada una de ellas a la Capa ATM para su transmisión. También le indica a la Capa ATM que establezca un bit en el campo PTI de la última célula, de manera que se conserven

los límites de mensaje. Puede argumentarse que esta es una mezcla incorrecta de capas de protocolo, pues la capa AAL no debería estar usando bits de la cabecera de la Capa ATM. El hacerlo viola el principio más básico de la ingeniería de protocolos y sugiere que las capas debieron diseñarse de manera diferente.

La ventaja principal del AAL5 sobre el AAL3/4 es una eficiencia mucho mayor. Si bien el AAL3/4 agregra solo 4 bytes por mensaje, también agrega 4 bytes por célula, reduciendo la capacidad de carga a 44 bytes, una pérdida del 8 por ciento en mensajes grandes. El AAL5 tiene un apéndice ligeramente más grande por mensaje (8 bytes) pero no tiene carga extra en cada célula.

La falta de números de secuencia en las células se compensa con la suma de comprobación más grande que puede detectar células perdidas , mal introducidas o faltantes sin necesidad de números de secuencia.

En la comunidad internet se espera que la manera normal de comunicarse con las redes ATM será el transporte de paquetes IP en campo de carga útil del AAL5. En el RFC 1483 y el RFC 1577 se estudian varios temas relacionados con este enfoque.

2.3 Capa ATM

La Capa ATM se orienta a conexiones, tanto en términos del servicio que ofrece como de la manera que opera internamente. El elemento básico de la Capa ATM es el circuito virtual (llamado con frecuencia canal virtual). Un circuito virtual normalmente es una conexión de un origen a un destino. Los circuitos virtuales son unidireccionales, pero puede crearse un par de circuitos al mismo tiempo. Ambas partes del par se direccionan al mismo identificador, de modo que un circuito virtual de hecho es dúplex integral. Sin embargo, la capacidad del canal y otras propiedades pueden ser diferentes en las dos direcciones, e inclusive pueden ser cero en una de ellas.

La Capa ATM es inusual para un protocolo orientado a conexiones en el sentido de que no proporciona acuses de recibo. La razón de este diseño es que ATM se diseñó para usarse en redes de fibra óptica, que son altamente confiables. Se consideró adecuado dejar el control de errores a las capas superiores. Como sabemos el envío de acuses de recibo en la capa de enlace de datos o en la de red sólo es una optimización. Siempre es suficiente que la capa de transporte envíe nuevamente el mensaje completo si no se tiene un acuse de recibo a tiempo.

Es más, las redes ATM se usan con frecuencia para tráfico en tiempo real, como

audio y vídeo. Para este tipo de tráfico, la retransmisión ocasional de una célula mala es peor que simplemente ignorarla.

A pesar de su falta de acuse de recibo, la Capa ATM sí proporciona una garantía firme: las células que se envíen por un circuito virtual nunca llegarán fuera de orden. Se permite a la subred ATM descartar células si ocurren congestionamientos, pero en ninguna circunstancia puede reordenar las células enviadas por un circuito virtual. Sin embargo, no se dan garantías sobre el orden si un terminal envía células por diferentes circuitos virtuales.

La Capa ATM reconoce una jerarquía de conexión de dos niveles que es visible a la capa de transporte. A lo largo de cualquier trayectoria de transmisión de un origen dado a un destino dado, un grupo de circuitos virtuales puede agruparse en lo que se llama una trayectoria virtual, como se representa en la figura 2.9. Conceptualmente una trayectoria virtual es como un haz de pares trenzados de cobre: al reenrutarlo, todos los pares (circuitos virtuales) se reenrutan juntos.

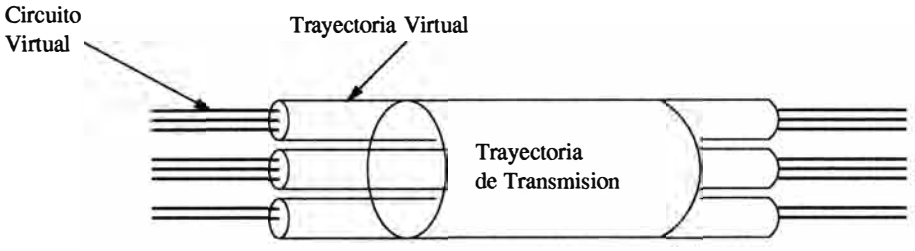

Figura 2.9: Trayectoria Virtual

2.3.1 Formatos de célula

En la recomendación CCITT I.361 se describe en detalle la codificación de células ATM. La estructura de célula que fue finalmente seleccionada por la CCITT contiene 48 octetos de campo de información y 5 octetos de cabecera. Los octetos son enviados en orden creciente, empezando por el primer octeto de la cabecera. Dentro de un octeto, los bits son enviados en orden decreciente, empezando en bit de mayor peso.

En la Capa ATM se distinguen dos interfaces: la UNI (User Network Interface, interfaz usuario-red) y la NNI (Network-Network Interface,interface red-red) La primera define el límite entre un terminal y una red ATM. La última se aplica a líneas entre dos conmutadores ATM. En ambos casos, las células consisten en una cabecera de 5 bytes seguida de una carga útil de 48 bytes, pero las dos cabeceras son ligeramente diferentes. Las cabeceras, como las definieron en el foro ATM, se

ilustran en la figura 2.10. Las células se transmiten comenzando por el byte más a la izquierda, y por el bit más a la izquierda de cada byte.

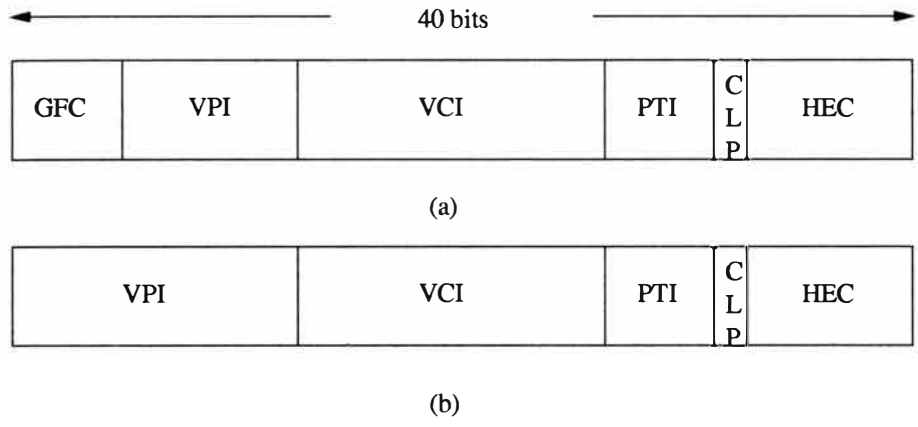

Figura 2.10: (a)Cabecera célula UNI. (b) Cabecera célula NNI

El campo GFC está presente sólo en las células entre un terminal y la red; es sobrescrito por el primer conmutador al que llega, por lo que no tiene significado de terminal a terminal, y no se entrega al destino. Originalmente se pensó que este campo tendría alguna utilidad para el control de flujo o de prioridad entre los terminales y las redes, pero no hay valores definidos para él, y la red lo ignora.

El campo VPI es un entero pequeño que selecciona una trayectoria virtual en particular. De la misma manera, el campo VCI selecciona un circuito virtual en particular en la trayectoria virtual seleccionada. Dado que el VPI tiene 8 bits (caso UNI) y el campo VCI tiene 16 bits, en teoría un host puede tener hasta 256 haces de VC, conteniendo cada uno hasta 65536 circuitos virtuales. En realidad, la cantidad disponible de ambos es un poco menor, pues algunos VCI se reservan para funciones de control, como el establecimiento de circuitos virtuales.

El campo PTI define el tipo de carga útil que contiene la célula. Aqui los tipos de célula son proporcionados por el usuario, pero la información de congestionamientos es proporcionada por la red. En otras palabras una célula enviada con PTI 000 podría llegar con 010 para avisar al destino que hay problemas en el camino.

El bit CLP puede ser establecido por un terminal para distinguir entre el tráfico de alta prioridad y el de baja prioridad. Si ocurre un congestionamiento y deben descartarse células, los conmutadores primero intentan descartar las que tienen el CLP establecido en 1 antes de descartar cualquiera que lo tenga establecido en 0.

Por último el campo HEC como se mencionó anteriormente es una suma de comprobación de la cabecera; no verifica la carga útil. Un código Hamming de un número de 40 bits solo requiere 5 bits, por lo que, con 8 bits, puede usarse un código

más refinado. El escogido puede corregir todos los errores de un solo bit y puede detectar aproximadamente el 90% de los errores multibit. Numerosos estudios han demostrado que la gran mayoría de los errores de los enlaces ópticos son errores de un solo bit.

A continuación de la cabecera vienen 48 bytes de carga útil. Sin embargo, no todos los 48 bytes están disponibles para el usuario, pues algunos de los protocolos AAL ponen sus cabeceras y sus terminaciones dentro de la carga.

El formato NNI es igual al formato UNI, excepto que el campo GFC no está presente y esos 4 bits se usan para hacer que el campo VPI sea de 12 bits en lugar de 8.

2.3.2 Establecimiento de conexiones

ATM maneja circuitos virtuales tanto permanentes como conmutados. Los primeros siempre están presentes y pueden usarse a voluntad, como en las líneas arrendadas. Los segundos tienen que establecerse cada vez que se usan, como al hacer una llamada telefónica. En esta sección describiremos la manera en que se establecen los circuitos virtuales conmutados.

Técnicamente, el establecimiento de la conexión no es parte de la Capa ATM, sino que es manejado por el plano de control usando un protocolo ITU muy complicado llamado Q.2931. Sin embargo, el establecimiento de conexiones es manejado por la Capa ATM por lo que consideramos su descripción en esta parte.

Hay varias maneras de establecer una conexión. La normal es adquirir primero un circuito virtual para señalización, y usarlo. Para establecer tal circuito, células que contienen una solicitud se envían por la trayectoria virtual 0, circuito virtual 5. Si hay éxito, se abre un circuito virtual nuevo por el que pueden enviarse y recibirse solicitudes y respuestas de establecimiento de conexión.

La razón de este procedimiento de establecimiento de dos pasos es que de esta manera el ancho de banda reservado para el circuito virtual 5 puede mantenerse extremadamente bajo. También se proporciona una manera alterna de establecer circuitos virtuales. Algunas portadoras pueden permitir que los usuarios tengan trayectorias permanentes entre destinos predefinidos, o permitirles establecerlos dinámicamente.

El establecimiento de un circuito virtual usa los seis tipos de mensajes listados en la figura 2.11. Cada mensaje ocupa una o más celulas y contiene el tipo de mensaje, la longitud y algunos parámetros. Los mensajes pueden ser enviados por

una estación final a la red o por la red (generalmente en respuesta a un mensaje de otra estación final) a una estación final. También existen varios otros mensajes de informe de estado y de errores, pero no se muestran aquí.

Mensaje
ESTABLECER
LLAMADA EN PROCESO
CONEXION
CONEXION RECONOCIDA
LIBERACION
LIBERACION COMPLETA

Figura 2.11: Mensajes usados para establecer y liberar conexiones

El procedimiento normal para establecer una llamada es que un host envíe un mensaje de ESTABLECER por un circuito virtual especial. La red responde con LLAMADA EN PROCESO para reconocer la recepción de la solicitud. A medida que el mensaje de ESTABLECER se propaga hacia el destino, es reconocido en cada salto por un mensaje de LLAMADA EN PROCESO.

Cuando el mensaje de ESTABLECER finalmente llega a su destino, el terminal de destino puede responder con CONEXION para aceptar la llamada. La red envía entonces un mensaje de CONEXION RECONOCIDA para indicar que se ha recibido el mensaje de CONEXION. A medida que el mensaje de conexión se propaga de regreso al originador, cada conmutador que lo recibe lo reconoce con un mensaje de CONEXION RECONOCIDA. Esta secuencia de eventos se presenta en la figura 2.12(a).

La secuencia para liberar un circuito virtual es sencilla. El terminal que desea colgar simplemente envía un mensaje de LIBERACION, que se propaga al otro extremo y causa que el circuito se libere. En cada salto a lo largo del camino se reconoce el mensaje, como se muestra en la figura 2.12(b).

A fin de establecer una conexión con un destino, es necesario especificar el destino, incluyendo su dirección en el mensaje de ESTABLECER. Las direcciones ATM tienen tres formas. La primera es de 20 bytes de longitud y se basa en direcciones OSI. El primer byte indica en cuál de tres formatos está la dirección. En el primer

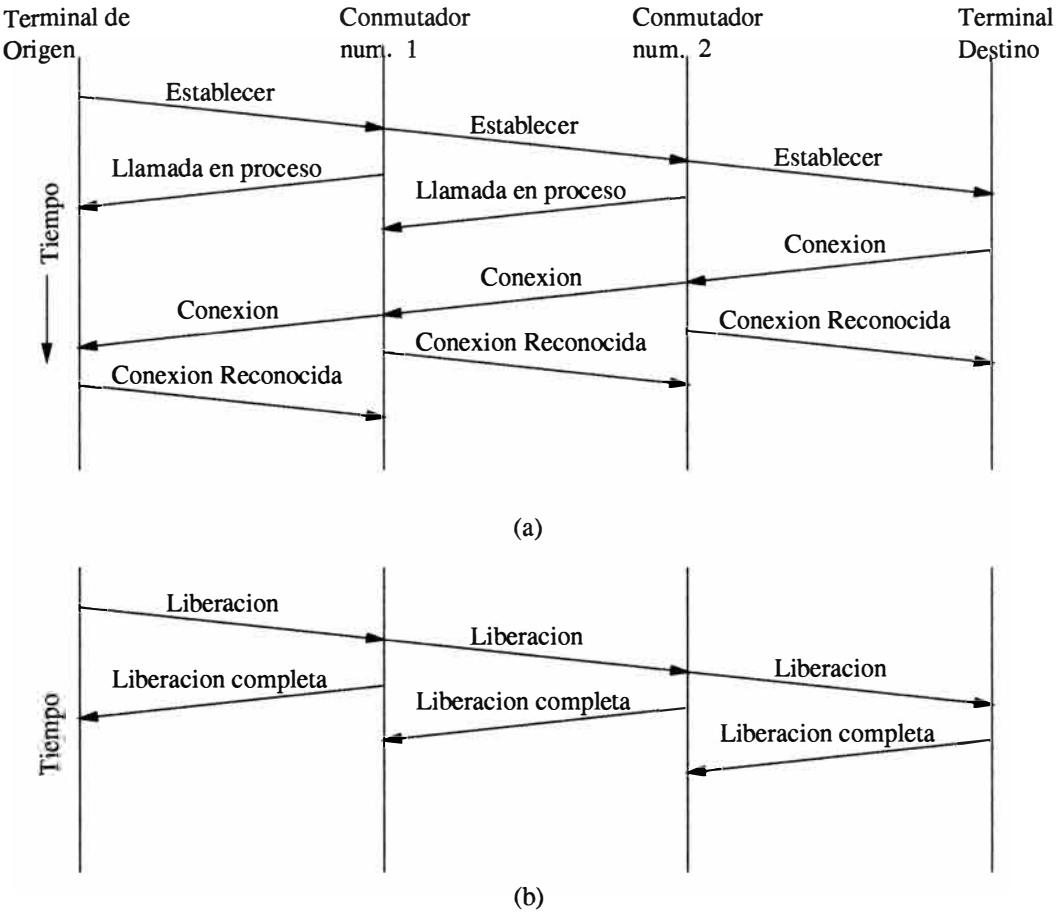

Figura 2.12: (a) Establecimiento de una conexión en una red ATM. (b) Liberación de la conexión

formato, los bytes 2 y 3 especifican un país, y el byte 4 da el formato del resto de la dirección, que contiene una autoridad de 3 bytes, un dominio de 2 bytes, un área de 2 bytes y una dirección de 6 bytes, más algunos otros elementos. En el segundo formato, los bytes 2 y 3 designan a una organización internacional en lugar de un país. El resto de la dirección es igual que en el formato 1. De manera alterna, también se permite una forma más vieja de direccionamiento (CCITT E.164) que usa números telefónicos ISDN decimales de 15 digitos.

2.3.3 Categorías de servicio ATM

Después de un buen tiempo de prueba y error , ya para la especificación 4.0 del ATM estaban claros los tipos de tráfico que transportaban las redes ATM y los tipos de servicios requeridos por el cliente. En consecuencia, el estándar se modificó para listar explícitamente las categorías de servicio comúnmente usadas, a fin de permitir a los proveedores de equipo optimizar sus interfaces y los conmutadores para algunas o todas estas categorías. Las categorías de servicios seleccionadas se listan en la

figura 2.2.

La clase CBR(tasa de bits constante) pretende la simulación de un medio de transmisión físico (solo que a un costo mayor). Los bits son enviados del origen al destino sin comprobación de errores, control de flujo, ni ningún otro proceso. No obstante, esta clase es esencial para hacer una transición suave entre el sistema telefónico actual y los sistemas B-ISDN del futuro, ya que en los canales PCM de grado de voz, los circuitos T1 y la mayor parte del resto del sistema telefónico usa la transmisión de bits síncrona de tasa constante. Con la clase CBR puede transportarse directamente todo este tráfico a través de un sistema ATM. La CBR también es adecuada para todas las demás cadenas interactivas (es decir , en tiempo real) audio y vídeo.

La siguiente clase , la VBR (tasa variable de bits), se divide en dos subclases, la de tiempo real (RT-VBR) y la de tiempo no real (NRT-VBR), respectivamente. La RT-VBR es para servicios que tienen tasas de bits variables en combinación con requisitos muy estrictos de tiempo real, como el vídeo comprimido interactivo (por ejemplo videoconferencias).Debido a la manera en que funcionan el MPEG y otros esquemas de compresión, con un marco base completo seguido de una serie de diferencias entre el marco actual y el marco base, la tasa de transmisión varía intensamente en el tiempo. A pesar de esta variación, es esencial que la red ATM no genere ninguna fluctuación en el patrón de llegada de células, ya que esto causaría que la imagen apareciese inestable. En otros palabras, tanto el retardo medio de las células como la variación del retardo de las células deben estar cuidadosamente controlados. Por otra parte, aquí es tolerable un bit o célula perdido de vez en cuando y lo mejor es ignorarlo.

La otra subclase VBR es para tráfico en el que la entrega a tiempo es importante, pero, la aplicación puede tolerar una cierta cantidad de fluctuación. Por ejemplo, el correo electrónico multimedia normalmente se almacena temporalmente en el disco local del receptor antes de presentarse en pantalla, por lo que cualquier variación en los tiempos de entrega de las células se eliminaría antes de visualizarse el correo electrónico.

La categoría de servicio ABR (tasa de bits disponible) se diseñó para trabajo en ráfagas cuya gama de ancho de banda se conoce aproximadamente. Un ejemplo típico podría ser su uso en una compañía que conecta sus oficinas mediante un conjunto de líneas arrendadas. Normalmente, la compañía puede seleccionar entre instalar suficiente capacidad para manejar la carga pico, lo que significa que algunas

líneas están ociosas parte del día, o instalar la suficiente capacidad para la carga mínima, lo que conduce a congestionamientos durante la parte más atareada del día. El uso del servicio ABR evita tener que comprometerse con un ancho de banda fijo.

Una aplicación que use ABR, especifica una PCR a ser usada y una MCR (tasa mínima de células) que requiere. La red reserva recursos de tal manera que todas las aplicaciones ABR reciben al menos su capacidad ABR. Cualquier capacidad no usada es compartida en un modo controlado y de acuerdo a reglas de contrato entre las fuentes ABR. El mecanismo ABR usa retroalimentación explícita hacia las fuentes para asegurar que la capacidad contratada se cumpla. Cualquier capacidad no usada por ABR permanece disponible para tráfico UBR.

Por último llegamos a la UBR (tasa de bits no especificada), que no hace promesas y no realimenta información sobre los congestionamientos. Esta categoría se adapta bien al envío de paquetes IP, puesto que IP tampoco hace promesas respecto a la entrega. Se aceptan todas las células UBR y , si sobra capacidad, también se entregan. Si ocurren congestionamientos, se descartan las células UBR sin realimentación al transmisor y sin esperar que el transmisor reduzca su velocidad. Como podemos observar este servicio es ideal para aplicaciones que puedan tolerar retardos variables y pérdida de algunas células, además de establecer su propio control de errores y de flujo.

2.4 Capa Física ATM

La Capa física de ATM cubre en términos generales las capas física y de enlace de datos del modelo OSI, siendo la subcapa dependiente del medio físico funcionalmente similar a la capa física de OSI y teniendo la Subcapa de Convergencia (TC) la funcionalidad de enlace de datos. No hay características de capa física en ATM. En su lugar, las células de ATM son transportadas por SONET, FDDI y otros sistemas de transmisión.

2.4.1 Subcapa Dependiente del Medio Físico

Esta subcapa es dependiente de la correcta transmisión y recepción de bits en el medio físico apropiado. Esta subcapa debe garantizar una apropiada reconstrucción temporizada de bit en el receptor. Sin embargo la entidad punto de transmisión será responsable de la inserción de información de temporización de bit y codificación de línea. Las subcapas de medio físico aplicables se encuentran especificadas por las

recomendaciones CCITT G.703, G.957 y por el Forum ATM.

2.4.2 Subcapa de Convergencia de Transmisión

En esta subcapa, los bits son aún reconocidos, a medida que llegan de la subcapa PMD.

La primera función despues de la reconstrucción de bits es la adaptación al sistema de transmisión usado. Los sistemas de transmisión posibles están basados en G.709 SDH, G.703 PDH o en células. Las células son encajadas dentro del sistema de transmisión de acuerdo a un mapeo estandarizado. Aquí también el Forum ATM ha adicionado FDDI como una opción para la interface usuario-red (UNI).

Esta subcapa es también responsable de la generación del campo HEC de cada célula en el transmisor y su verificación en el receptor.

Finalmente esta subcapa debe asegurar la inserción y supresión de células no asignadas, para adaptar la tasa útil al campo de carga (payload) del sistema de transmisión. Esta función es llamada desacoplo de tasa de células.

Capítulo 3

Linux ATM

3.1 El Sistema Operativo Linux

El sistema operativo linux es un clone de Unix de distribución gratuita con una gran base de soporte. El total del kernel y prácticamente todas las aplicaciones necesarias para construir un sistema Unix completo estan disponibles en forma libre así como el código fuente bajo GNU (Licencia Pública General).

3.1.1 Sockets API BSD

La interface de red TCP/IP en Linux esta basada en Sockets API BSD, lo cual es un estándar en la mayoría de sistemas operativos Unix. Este API forma la base de los Sockets API ATM . La interface de programación basada en sockets está bien documentada en muchos textos de programación en redes [8].

La interface socket es una interface "de conexión virtual"muy genérica. Esta tiene varias propiedades, dependiendo del protocolo usado, estructuras de datos específicas a ese protocolo y más, ver [10].

La interface socket BSD provee un medio para crear una asociación entre una fuente y destino(s) en un protocolo específico, interface u otro medio de comunicación. Tanto datos orientados a la conexión como sin conexión pueden ser enviados a través del socket.

3.1.2 Espacio Kernel vs Espacio Usuario

Sin ir muy lejos dentro de los detalles de diseño del sistema operativo, existen dos clases de ambiente en los cuales puede operar el software, el espacio Kernel y el espacio Usuario,ver [5]. El espacio kernel es un modo de operación privilegiado y es usado por código compilado en el kernel o cargado como módulo despues del proceso de arranque inicial del sistema. El espacio de usuario esta orientado a aplicaciones ejecutadas por el usuario. Estas aplicaciones pueden ser demonios, aplicaciones interactivas o en batch. Los manejadores de dispositivos son siempre ejecutados en

el espacio kernel, debido a que estos necesitan acceder a interrupciones de hardware y otras funciones de bajo nivel no disponibles en el espacio usuario.

3.2 Driver Linux ATM

Desde comienzos de 1995, el soporte ATM está siendo desarrollado para el sistema operativo Linux. El contexto bajo el cual el desarrollo esta siendo conducido es el proyecto "Web sobre ATM"en el "Laboratoire de Réseaux de Communication(LRC). El desarrollo es abierto , así como común en el entorno Linux, habiendo gente que ha contribuido en la elaboración de drivers e implementaciones de protocolo (por ejemplo:LANE, CLIP, Arequipa, MPOA).

3.2.1 Características distintivas soportadas por Linux ATM

El paquete Linux ATM está en constante desarrollo, debido a esto la lista de características es cambiante en el tiempo. Lo que sigue es una lista de características, que son mantenidas actualizadas en el documento README de la distribución. Esta información corresponde a la versión 0.59 de la distribución Linux ATM.

- PICs(Tarjetas Interface Físicas) soportadas: Efficient Networks EN155p-MF/U5, SMC ATM Power 155, Rolf Fiedlers TNET A1570 board, UniNET1570, Zeinet ZN1221/ZN1225 155 Mbps ATM adapter,Fore PCA-200E.

Adicionalmente una pseudo interface ATM llamada "ATM sobre TCP"(ATM-TCP) es proveída , esta interface permite realizar pruebas de los sockets ATM, sin la necesidad de contar con una PIC ATM instalada, basada en la encapsulación de células ATM en el payload de los PDUs TCP y su transporte en una conexión TCP sobre IP.

Entre los tipos de conexión que son soportados por el driver tenemos:

- PVCs : VCCs UBR bidireccionales punto a punto.
- SVCs : VCCs UBR bidireccionales punto a punto.

El tipo de señalización usada para los SVCs es UNI 3.0 o 3.1 , esto es seleccionado usando los flags del compilador del demonio de señalización (atmsigd).

En caso de información más detallada se recomienda consultar el documento **USAGE** que acompaña la distribución.

Dentro de otras características que soporta tenemos:

- En el uso directo de sockets ATM, dos tipos de AAL estan disponibles: "raw"AAL0 y raw AAL5.

- IP clásico sobre ATM tanto usando encapsulación NULL o LLC/SNAP. Un servicio de ATMARP tal como esta definido en RFC 1577(tanto cliente como servidor).

- Un cliente y un servidor LANE (incluyendo todos sus módulos LES, LEC, LECS y BUS) son proveídos también en esta distribución.

- Arequipa : Aplicación con petición de IP sobre ATM, es un protocolo que permite establecer una "conexión"IP entre dos entidades IP de dos elementos conectados vía una red ATM.

- ANS : Servicio de nombres ATM. Un servicio para resolver nombres de computadores leíbles a direcciones ATM NSAP.

3.2.2 Hardware de la plataforma ATM utilizada

El hardware escogido para este proyecto consiste en tres tarjetas ATM FORE Runner PCA-200E de 155 Mbps, además de tarjetas de red ethernet(las cuales hacen uso de la facilidad de ATM sobre TCP).

El driver para dichas tarjetas se encuentra disponible en internet, siendo fácil su incorporación al kernel y a la distribución ATM.

3.2.3 Estructura interna de los módulos Linux ATM

En la figura 3.1 se muestra la pila del protocolo Linux ATM junto a la pila del protocolo IP sobre Ethernet. Esta es un figura de referencia, en un sistema real los protocolos disponibles dependen del hardware instalado.

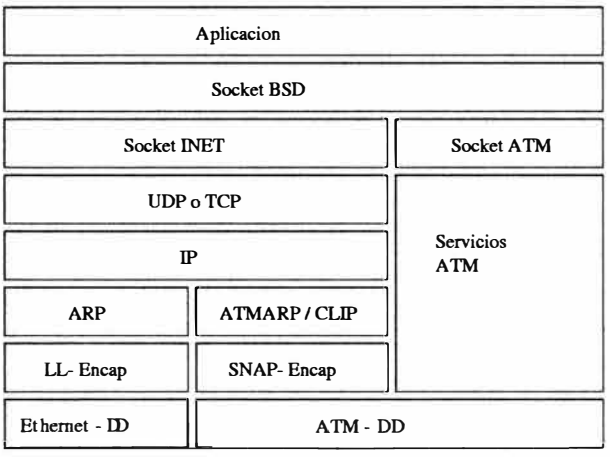

Figura 3.1: Configuración de referencia de protocolo Linux ATM

Esta no es la misma estructura que los módulos usados para operar Linux ATM y

las fuentes para las extensiones de kernel, pero muestra como las extensiones ATM son integradas dentro de la interface socket Linux/BSD. En la cima tenemos la Aplicación, que emplea la interface socket BSD para comunicarse. En esta configuración, la aplicación puede escoger abrir tanto un socket INET o un socket ATM. En el nivel de IP , se puede eligir usar tanto interface Ethernet o la interface ATMARP/CLIP para la transmisión de paquetes IP. La casilla inferior indica el DD (driver de dispositivo) de la PIC instalada.

En el sistema operando, la parte compleja de funcionalidad del protocolo es desarrollada en el espacio usuario por demonios. En el sistema Linux ATM principalmente encontramos los siguientes demonios:

- `atmsigd` : provee soporte de señalización en la red, usando el PVC identificado por VPI=0 y VCI=5.

- `atmarpd` : provee soporte cliente y servidor para la funcionalidad ATMARP.

- `ilmid` : es usado para configurar las direcciones ATM combinando la información de dirección local con la parte de dirección de red, proveída por un switch ATM. Por defecto , el PVC identificado por VPI=0 y VCI=16 es usado por la ILMI.

En la figura 3.2 , los demonios atmsigd, ilmid y atmarpd son mostrados con sus interfaces al kernel. La casilla SVC* es un socket SVC especial usado para intercambiar información de señalización entre el kernel y los demonios.

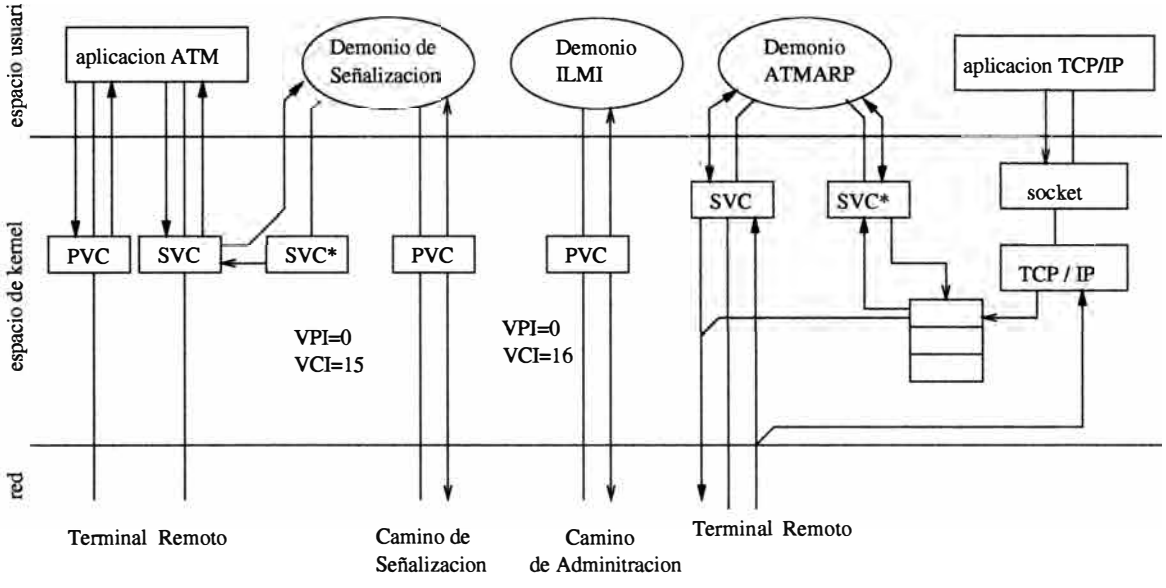

Figura 3.2: Estructura de Demonios Linux ATM

El kernel mantiene un tabla `ATMARP` con información parcial, la cual es usada para el envío de paquetes sobre VCCs existentes y en el caso que fuera necesario setear

un VCC a una dirección ATM conocida usando el demonio `atmarpd`. Este demonio también recibe transmisiones IP, debido a que en la implementación, ATMARP es una capa entre la capa IP y la capa de encapsulado LLC/SNAP.

3.3 Interface de programación de aplicaciones ATM (API-ATM)

El Forum ATM define las especificaciones API para servicios nativos ATM, más no define una interface para sockets BSD. Debido a que la API ATM Linux esta basada en sockets BSD, ésta no conforma directamente las especificaciones del Forum ATM.

La semántica y filosifía de diseño de la interface socket BSD fue tratada de seguir lo más cerca posible para el diseño de la presente API. Las llamadas al sistema y sus parámetros tienen generalmente el mismo significado en la implementación socket ATM.

3.3.1 Estructuras de direcciones

Para hacer posible el uso de ATM por medio de una interface socket común, fue necesario definir nuevas estructuras de direcciones. Así dos nuevas familias de direcciones, AF_ATMPVC y AF_ATMSVC fueron creadas dentro del API.

La información dada en estas estructuras vía la primitiva socket, especifíca el SAP (punto de acceso a servicio) remoto, donde se intenta establecer una conexión o un SAP local, cuando se está a la escucha de llamadas entrantes.

```
struct sockaddr_atmpvc {
    unsigned short  sap_family; /* address family, AF_ATMPVC  */
    struct {                /* PVC address */
        short   itf;        /* ATM interface */
        short   vpi;        /* VPI (only 8 bits at UNI) */
        int     vci;        /* VCI (only 16 bits at UNI) */
    }       sap_addr;       /* PVC address */


struct sockaddr_atmsvc {
    unsigned short  sas_family; /* address family, AF_ATMSVC */
    struct {                    /* SVC address */
        unsigned char   prv[ATM_ESA_LEN];/* private ATM address */
        char        pub[ATM_E164_LEN+1]; /* public address (E.164) */
                    /* unused addresses must be bzero'ed */
        struct atm_blli *blli;      /* local SAP, low-layer information */
        struct atm_bhli bhli;       /* local SAP, high-layer information */
    }           sas_addr;       /* SVC address */
    struct atm_trafprm  sas_txtp;   /* TX traffic parameters */
    struct atm_trafprm  sas_rxtp;   /* RX traffic parameters */
```

```
};

struct atm_blli {
    unsigned char l2_proto; /* layer 2 protocol */
    union {
    ...
    } l2;
    unsigned char l3_proto; /* layer 3 protocol */
    union {
    ...
    } l3;
    struct atm_blli *next;  /* next BLLI or NULL (undefined when used in */
                /* atmsvc_msg) */
    };


struct atm_bhli {
    unsigned char hl_type;  /* high layer information type */

    };
```

La parte del kernel de ATM en Linux y el demonio de señalización usan esta información cuando se realiza la apertura de conexiones.

3.3.2 Direccionamiento de PVCs

En el caso de la estructura de dirección para conexiones PVC, se define la interface local y el identificador de conexión (VPI y VCI) a usarse en la respectiva interface.

Los formatos de representación textual para direcciones PVC son

1. *vpi.vci*
2. *itf.vpi.vci*

itf, *vpi*, y *vci* son números decimales no negativos no precedidos por ceros, caracteres especiales ? ó *. En el caso del primer formato, se asume que el número de interface corresponde a cero.

3.3.3 Direccionamiento de SVCs

Para el caso de conexiones SVC en formación, se emplea la información dada en las respectivas estructuras mostradas anteriormente, para llenar los campos necesarios del mensaje de señalización ESTABLECER, cuando se intenta la conexión a un terminal remoto. El socket en estado de escucha creado con estas estructuras

es notificado cuando un mensaje de ESTABLECER con los campos apropiados es recibido desde la red.

El direccionamiento de SVCs ocurre en tres niveles:

- Selección de un gateway de conmutación de conexiones ATM entre una red pública y una red privada.

- Selección de un terminal final dentro de una red pública o privada

- Selección de una especificación de servicio en el terminal final seleccionado.

Las redes públicas emplean numeración E.164 para direccionamiento. Las redes privadas emplean direcciones NSAP privadas basadas en el ATM Forum. En el caso del API ambos tipos de direcciones son tratadas como direcciones de red privadas.

Para poder seleccionar un terminal, las siguientes direcciones son requeridas, tomando como referencia la figura 3.3:

- Si el destino se encuentra en una red privada : la dirección privada (A→B)

- Si el destino se encuentra en una red pública(usando direcciones E.164): la dirección pública (A→D, D→E)

- Si el destino se encuentra en una red privada, la cual es alcanzada via una red pública usando direcciones E.164 : las direcciones privada y pública (A→F→G)

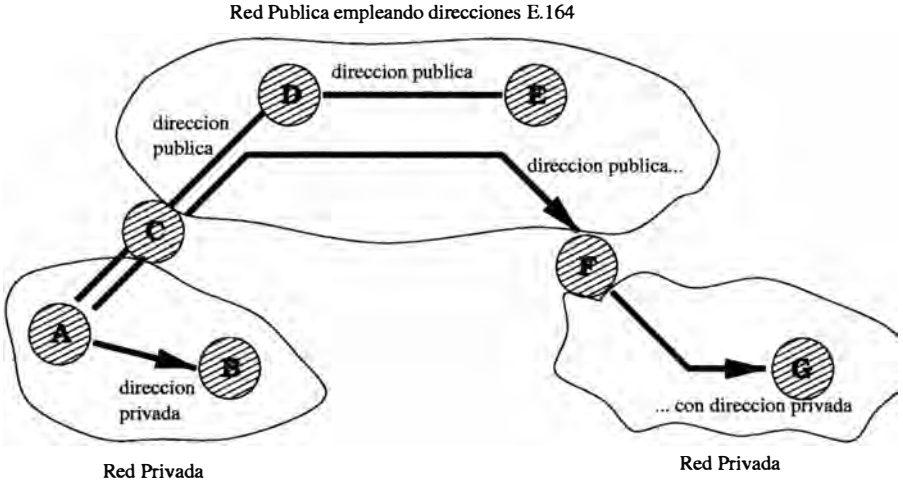

Figura 3.3: Direccionamiento de SVCs

El servicio en un terminal final es identificado mediante la especificación o bien de protocolo de capa-inferior (capas OSI 2 ó 3), o protocolo de capa superior, de ambos o ninguno de ellos.

La estructura de direccionamiento SVC mostrada anteriormente se encuentra definida en `/usr/include/linux/atm.h`. De ésta, tenemos los campos :

- `prv`: dirección privada ATM, especifica una dirección NSAP de un terminal en

una red privada. Su longitud es siempre **ATM_ESA_LEN** (veinte) bytes.

- **pub**: corresponde a una dirección E.164. La dirección pública es una cadena ascii de **ATM_E164_LEN** dígitos decimales

- **blli** y **bhli**: identifican el punto local de acceso de servicio (SAP),en general : la aplicación llamante o llamada o una parte funcional de la última. Ambos campos son opcionales, ver secciones 2.4.3 y 2.4.4 de [2] para mas detalles.

3.3.4 Descriptores de QOS

La interface Linux ATM maneja los requerimientos de Calidad de Servicio para las distintas conexiones mediante descriptores de QoS. Estos se encuentran codificados en una estructura de datos del tipo **struct atm_qos**, la cual está definida en **/usr/include/linux/atm.h**.

```
struct atm_qos {
    struct atm_trafprm txtp;    /* parameters in TX direction */
    struct atm_trafprm rxtp;    /* parameters in RX direction */
};
```

txtp y **rxtp** son los parámetro de tráfico en las direciones de envio y recepción, respectivamente. Su estructura es descrita en los siguientes puntos.

- Parámetros de tráfico

Los parámetros de tráfico son adicionados al descriptor de QoS simplemente asignando valores a los campos correspondientes. Estos parámetros se encuentran codificados en una estructura de datos del tipo **struct atm_trafprm** (definida en **/usr/include/linux/atm.h**) la cual contiene los siguientes campos :

```
struct atm_trafprm {
    unsigned char   class;      /* traffic class (ATM_UBR, ...) */
    int     max_pcr;            /* maximum PCR in cells per second */
    int     min_pcr;            /* minimum PCR in cells per second */
    int     max_cdv;            /* maximum CDV in microseconds */
    int     max_sdu;            /* maximum SDU in bytes */
};
```

class especifica la clase de tráfico, indicando propiedades generales de tráfico. **min_pcr** y **max_pcr** indican el valor de tasa pico de células, en células por segundo. **max_cdv** indica la variación de retardo de célula en microsegundos. **max_sdu** se refiere al tamaño e la unidad máxima de servicio de datos (SDU).

- Clases de Tráfico

Las siguientes clases de tráfico están definidas:

ATM_NONE no traffic in this direction

ATM_ANYCLASS accept any class

ATM_CBR constant bit rate (CBR)

ATM_UBR unassigned bit rate (UBR)

En el caso que no se especifique clase de tráfico, se asume **ATM_NONE** por defecto, en este caso todos los demás parámetros de tráfico son ignorados.

ATM_ANYCLASS es usado cuando se desea especificar SAPs SVC que sean compatibles con cualquier clase de tráfico requerida por la parte origen del llamado.

Dependiendo de la clase de tráfico, únicamente ciertos parámetros de tráfico son usados:

	ATM_NONE	ATM_CBR	ATM_UBR
max_pcr	–	Yes	–
min_pcr	–	Yes	–
max_cdv	–	Yes	–
max_sdu	–	Yes	Yes

3.3.5 Funciones de librería adicionales del API

Un conjunto de funciones es proveído para codificar, decodificar y comparar direcciones , además para convertir tasas de transmisión SDU, en tasas de transmisión por células.

Para poder usar esta librería, el archivo de cabecera `/usr/include/linux/atm.h` debe ser incluído. Además de eso, la librería libatm.a debe ser enlazada al momento de compilación.

En la presente tesis por ejemplo, hacemos uso de la función `text2atm` que convierte una representación textual de una dirección PVC o SVC en su correspondiente codificación binaria.

```
int text2atm(const char *text,struct sockaddr *addr,
             int length,int flags);
```

`text` apunta a una cadena terminada en carácter NULO, conteniendo la representación textual de la dirección. `addr` apunta a una estructura de datos suficientemente larga para almacenar la dirección de la estructura resultante, la cual puede ser tanto una `struct sockaddr_atmpvc` o una `struct sockaddr_atmsvc`. `length` indica la longitud de la estructura de datos. Esta conversión falla en el caso de no proveerse de suficiente espacio. `flags` es un conjunto de opciones de procesamiento:

```
#define T2A_PVC          1
#define T2A_SVC          2
#define T2A_UNSPEC       4
#define T2A_WILDCARD     8
#define T2A_NNI         16
#define T2A_NAME        32
```

`text2atm` retorna la longitud de NSAP en caso de éxito (0 si no existe parte NSAP, 160 si hay una, o algún entero intermedio en el caso de usar wildcards), un valor negativo en caso de falla.

En caso de éxito, `text2atm` inicializa los siguientes campos en PVCs: `sap_family` y todos los campos en `sap_addr`.

3.3.6 Fases API para sockets PVC

Aperturar un socket PVC significa agregar un punto final a una conexión que normalmente ya existe en la red o que es establecida por una tercera parte. El diagrama de estados en la figura 3.4 muestra el ciclo de vida de los sockets PVC.

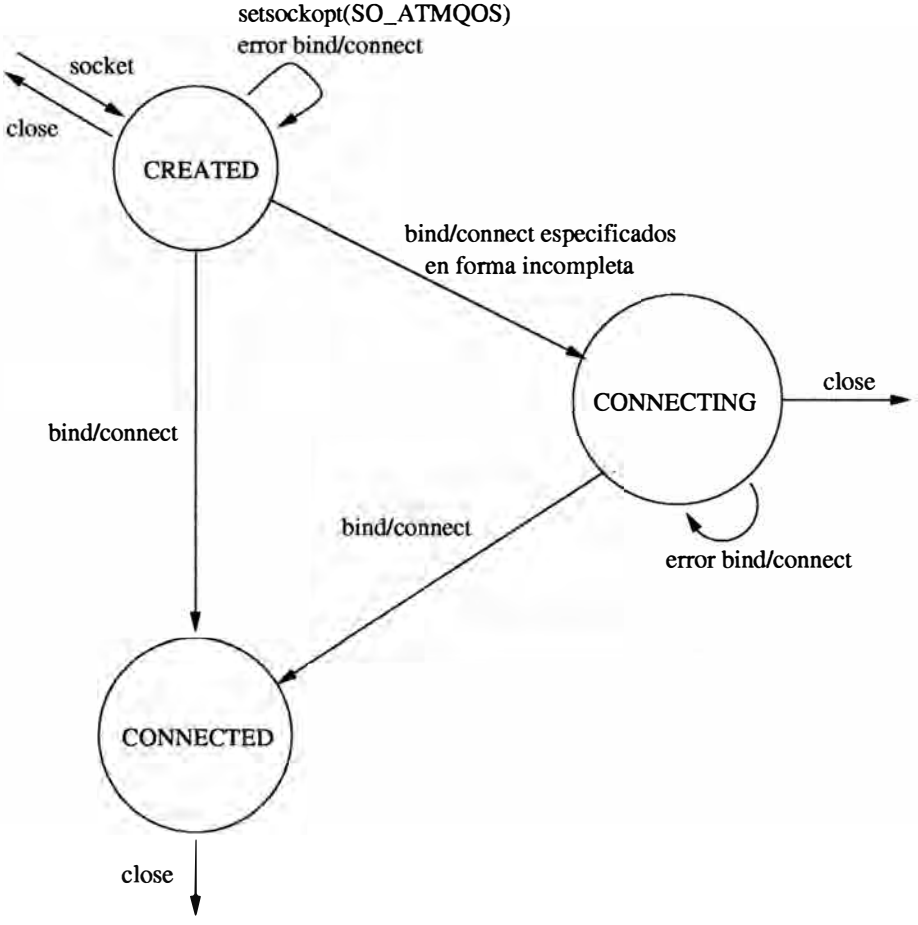

Figura 3.4: Apertura de sockets PVC

Los sockets PVC son creados por medio de la primitiva *socket*. Luego que los requerimientos de QOS son indicados, un descriptor de dirección es establecido, y las primitivas *bind* o *connect* (estas tienen idéntica funcionalidad cuando son empleadas con PVCs) son llamadas para abrir un VC. Si el VC no puede ser abierto, la operación puede ser reintentada, posiblemente despues de cambiar el descriptor de dirección y/o los parámetros de tráfico.

Una vez que un VC es establecido, se puede intercambiar datos hasta que el socket es cerrado por medio de la primitiva *close*. Es posible también cerrar el socket en cualquier estado.

3.3.7 Fases API para sockets SVC

Las SVCs son conexiones cuyo establecimiento en la red esta controlado por las partes comunicantes. Un terminal puede tanto aceptar conexiones entrantes (apertura pasiva) o puede intentar establecer una conexión (apertura activa) hacia otro terminal. La figura 3.5 muestra el ciclo de vida de una conexión activa.

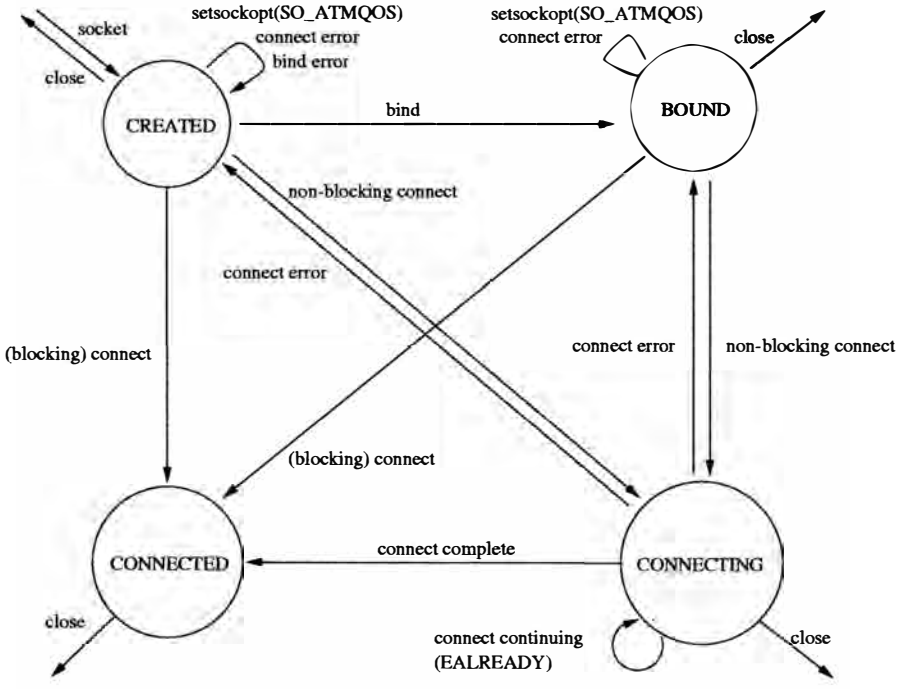

Figura 3.5: Apertura Activa de un socket SVC

El modo más común para establecer una conexión es crear el socket SVC con *socket*, especificar los requerimientos de QoS, preparar el descriptor de dirección, realizar la petición de ESTABLECER conexión con la primitiva *connect* bloqueante, esperar hasta que la red acepte (o rechaze) la conexión, intercambiar datos, y even-

tualmente cerrar el socket con *close.*

El punto final de conexión local puede ser identificado por una dirección local por medio de la primitiva *bind.* En el caso de usar la primitiva *connect* no bloqueante, el proceso continua ejecutándose mientras la conexión es establecida y este puede decidir incluso cerrar el socket prematuramente. Notar que existen dos transiciones a `connect error` a partir del estado CONNECTING. La elección de cual de ellas es tomada depende si el socket esta identificado (bound) o no.

La apertura pasiva de SVCs se muestra en la figura 3.6.

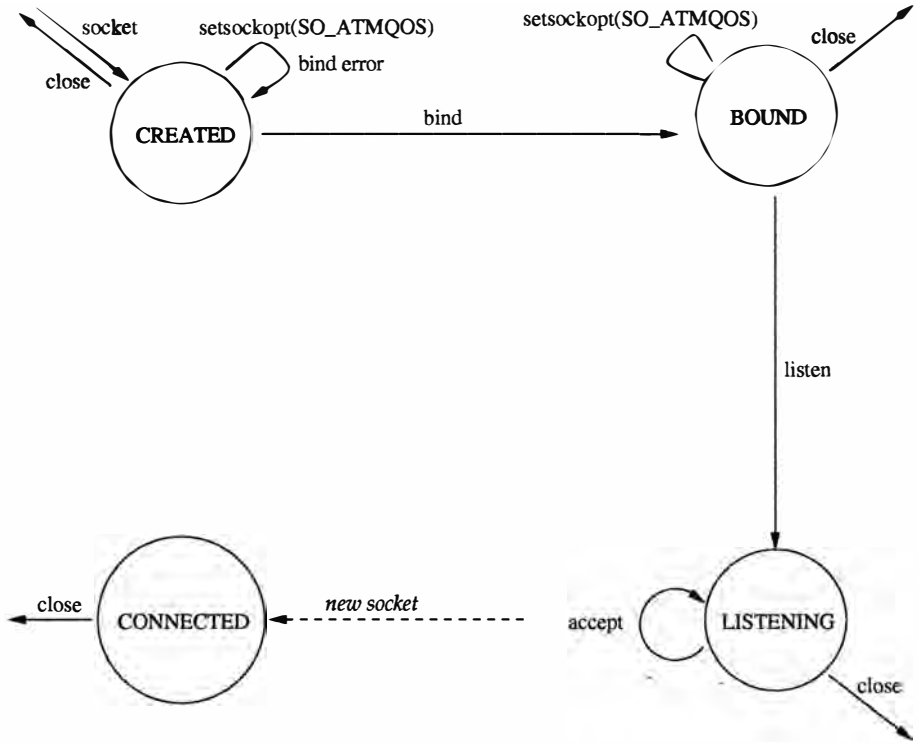

Figura 3.6: Apertura Pasiva de un socket SVC

Como es usual el socket es primero creado con la primitiva *socket* y los parámetros QOS son establecidos. Luego, el descriptor de dirección es configurado para especificar el tipo de conexiones que podrían ser atribuídas a este socket, y el socket es identificado con la primitiva *bind.* El SAP es registrado en la entidad de señalización local por medio de la primitiva *listen.*

Ahora el proceso puede ser bloqueado en la primitiva *accept* hasta que una petición de conexión sea recibida, o este puede hacer un sondeo usando las primitivas *accept* no bloqueante o *select.* Si una conexión entrante encaja dentro de las especificaciones del SAP, la primitiva *accept* tendrá éxito (o el socket se vuelve leíble de tal manera que *select* retorna y *accept* puede ser invocado). Si *accept* tiene éxito, un nuevo socket es creado para la conexión y el intercambio de datos puede ser iniciado.

El socket en modo pasivo y el (los) socket(s) usados para intercambiar datos pueden ser individualmente cerrados con *close*. Notar que cerrar el socket pasivo, únicamente remueve el registro SAP y no afecta las conexiones que hayan sido establecidas.

Capítulo 4

Plataforma ATM Implementada

4.1 Hardware

La figura 4.1 muestra la plataforma de red ATM implementada como parte del desarrollo de la presente tesis y sobre la cual se ha desarrollado la Aplicación de AudioConferencia.

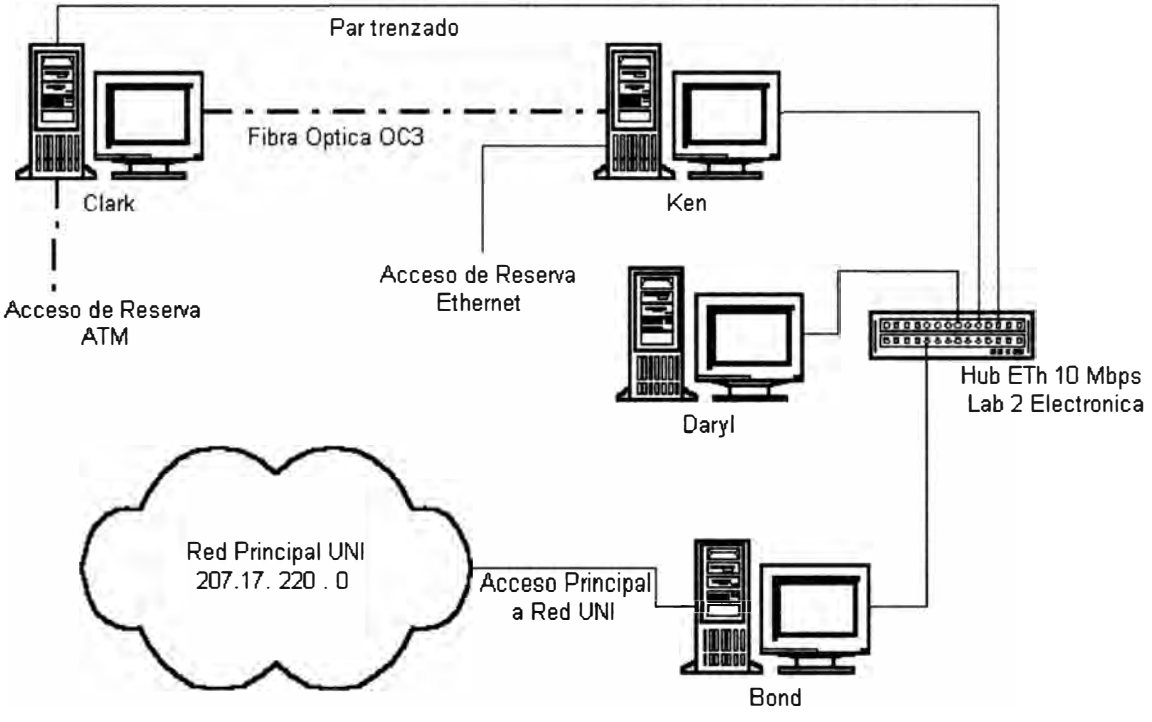

Figura 4.1: Plataforma de Red ATM

Como se puede observar en la figura, la plataforma de Red ATM consiste de una subred Ethernet(par trenzado) y una subred ATM (fibra óptica OC3). La composición principal de hardware de los distintos nodos es como sigue:

`Nodo Clark`: Computadora Pentium Celeron 333 MHz,64 MB en RAM, Tarjeta de sonido: Vibra Creative Sound Blaster 16 bits, Tarjetas de Red: 2 tarjetas ATM Fore PCA-200E 155 Mbps, 1 tarjeta Ethernet 10 Mbps.

`Nodo Ken`: Computadora Pentium Celeron 333 MHz,64 MB en RAM, Tarjeta de sonido: Vibra Creative Sound Blaster 16 bits, Tarjetas de Red: 1 tarjeta ATM Fore PCA-200E 155 Mbps, 2 tarjetas Ethernet 10 Mbps.

`Nodo Daryl`: Computadora Pentium MMX 233 MHz,32 MB en RAM, Tarjeta de sonido: Vibra Creative Sound Blaster 16 bits, Tarjetas de Red: 1 tarjeta Ethernet 10 Mbps.

`Nodo Bond`: Computadora Pentium S 100 MHz,24 MB en RAM, Tarjeta de sonido: Vibra Creative Sound Blaster 16 bits, Tarjetas de Red: 2 tarjetas Ethernet 10 Mbps.

4.2 Software

El software instalado en los distintos nodos es como sigue:

`Nodo Clark`: Sistema Operativo Linux RedHat 5.2, Kernel version 2.1.126, Distribución Linux-ATM version 0.52, Distribución Switch Virtual Linux ATM 0.5.

`Nodo Ken`: Sistema Operativo Linux RedHat 6.0, Kernel version 2.2.9, Distribución Linux-ATM version 0.59.

`Nodo Daryl`: Sistema Operativo Linux RedHat 6.0, Kernel version 2.2.9, Distribución Linux-ATM version 0.59.

`Nodo Bond`: Sistema Operativo Linux RedHat 6.0, Kernel version 2.2.9, Distribución Linux-ATM version 0.59.

4.3 Configuración y aspectos funcionales

Una de las características mas importantes de la plataforma de red ATM implementada, es su flexibilidad funcional, permitiendo realizar diversas configuraciones tanto para el desarrollo de pruebas de laboratorio, estudio y desarrollo de aplicaciones, a nivel de ATM nativo y de integración IP-ATM. En la presente tesis se ha considerado las siguientes configuraciones:

- Configuración Red Privada ATM.
- Configuración de arbitraje y gestión centralizada de aplicación AudioConferencia

4.3.1 Configuración red privada ATM

La figura 4.2 muestra la distribución funcional de los terminales de la plataforma ATM para una configuración de red privada ATM, en la cual se dispone de un punto de acceso (futuro) hacia la red ATM de la Universidad Nacional de Ingeniería.

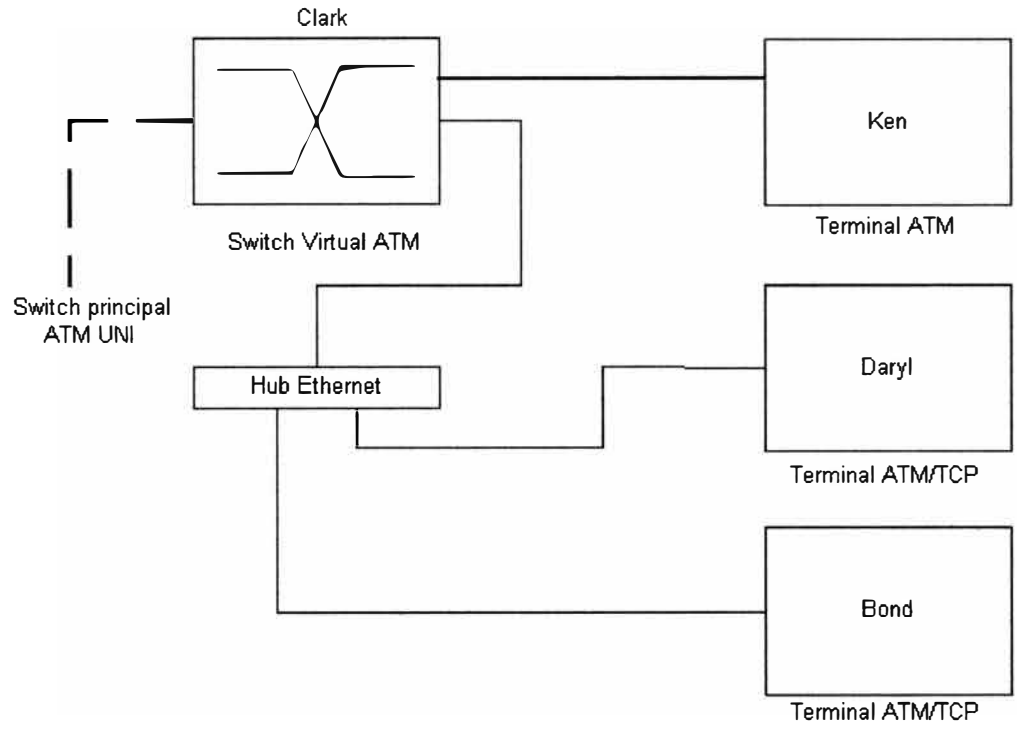

Figura 4.2: Configuración Red privada ATM

Como se aprecia en la figura, la configuración Red Privada ATM, hace uso del nodo Clark configurado como Switch Virtual Linux ATM, este switch virtual (conmutación por software) permite el manejo de 16 puertos ATM, donde cada puerto está constituido por una tarjeta interface de red, de modo que la capacidad máxima en puertos hardware estará limitada por el número de ranuras PCI disponibles en el terminal, pudiéndose emplear tanto tarjetas interface ATM como Ethernet.

Los nodos con interface Ethernet hacen uso de la facilidad ATM sobre TCP de la distribución Linux-ATM, de este modo la plataforma en su conjunto se observa como una única plataforma de red privada ATM; es necesario aclarar que tal como lo establece el documento [3], la facilidad de ATM sobre TCP permite la realización de pruebas de transmisión más no de pruebas que consideren el throughput o latencia del sistema, debido a que ésta es una emulación ATM.

4.4 Medidas de performance del soporte Linux ATM

En vista que la presente tesis está basada en el empleo de PVCs de transmisión de datagramas AAL5, determinaremos como medida de performance el throughput nominal y del soporte Linux ATM en función del tamaño SDU correspondiente a la transmisión de datagramas AAL5 en tráfico UBR, a fin de determinar los límites correspondientes y los valores de tasa de bits disponibles para el desarrollo de aplicaciones basadas en transmisión de datagramas UBR AAL5.

Nominalmente, el throughput es una medida de la cantidad de datos transferida por unidad de tiempo. Sin embargo, el número de bytes transferidos no sólamente incluye los datos a ser enviados, sino también incluye dentro del paquete una sobrecarga como son la cabecera y apéndice. La longitud de una célula ATM está fijada en 53 bytes: 48 bytes de datos y 5 bytes de cabecera. Adicionalmente, cada marco AAL5 posee un apéndice de 8 bytes al final. Si la longitud del marco no es divisible exactamente por 48, el protocolo ATM emplea bytes de relleno a ser añadidos para completar la última célula de un marco.

Supongamos que son F los bytes por SDU de información del nivel de aplicación a ser enviados. Cuando el SDU alcanza los niveles del protocolo ATM, se le añade un apéndice de 8 bytes. Los datos son luego divididos en células de 48 bytes con relleno, si fuera necesario, para llenar la última célula. Además cada célula posee un cabecera de 5 bytes adicionada. Luego el número total de células creadas será al menos:

$$\frac{(F+8+(F+8)mod48)\,bytes}{48bytes/célula} \tag{4.1}$$

Donde el operador *mod* permite obtener el número de bytes necesarios de relleno.

El número total de bytes transmitidos es:

$$\frac{53\,(F+8+(F+8)mod48)}{48} bytes \tag{4.2}$$

La capa física de la interface FORE PCA-200E ofrece una tasa de bits nominal de 155 Mbps. Así, el máximo troughput en la capa de aplicación estará en el límite de:

$$155\left(\frac{48F}{53(F+8+(F+8)mod48)}\right) Mbps. \tag{4.3}$$

La figura 4.3 representa en forma gráfica los valores de throughput obtenidos para distintos tamaños SDU.

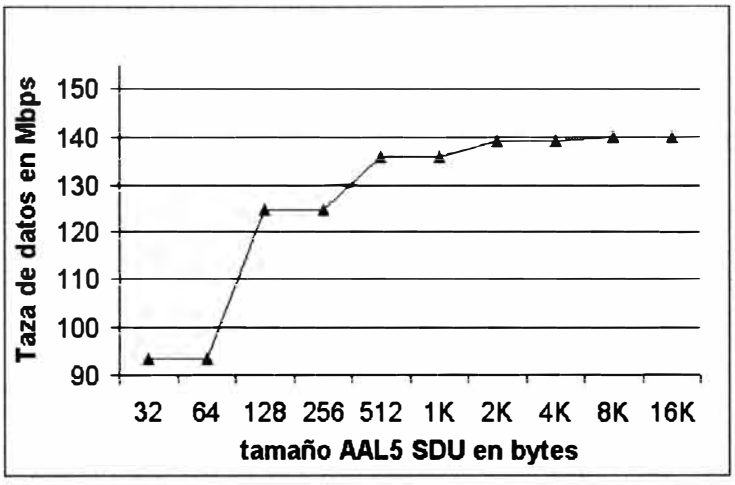

Figura 4.3: Valores teóricos de throughput para distintos tamaños de SDU

Experimentalmente, las medidas de performance fueron hechas en los nodos Ken y Clark, directamente conectados (back to back), empleando una version de la aplicación `ttcp` con extensión de sockets ATM, mediante esta aplicación se establece un circuito virtual AAL5, con calidad de servicio UBR, configurando un terminal en modo recepción y el otro en transmisión, respectivamente, con un tamaño SDU determinado, de acuerdo a las opciones de la aplicación, el terminal en modo recepción es inicialmente configurado e inicializado, encontrándose a la espera de paquetes de información ATM AAL5 a ser transmitidos a través de la conexión virtual establecida, por parte del otro terminal; la aplicación temporiza tanto el envío en el terminal transmisor, como la recepción en el terminal receptor de un número determinado de buffers (2048) de una longitud de 8192 bytes, en base a esta medida de tiempo visualiza los resultados de tiempo de transmisión o recepción, velocidad en Mbps, y número de bytes transmitidos o enviados respectivamente.

La figura 4.4 muestra el throughput para el caso de transmisiones unidireccionales de datagramas AAL5 con distintos tamanõs de SDU.

La medida experimental obtenida corresponde a transmisiones unidireccionales originadas a partir de un terminal o recepcionadas en otro, sin embargo existen gran cantidad de aplicaciones de la vida real que se basan en transmisiones bidireccionales, por lo cual es necesario obtener una medida de performance para los casos en cuales se desarrolle una aplicación de tales características sobre el entorno Linux ATM, dichas medidas de performance son obtenidas por medio de una configuración de

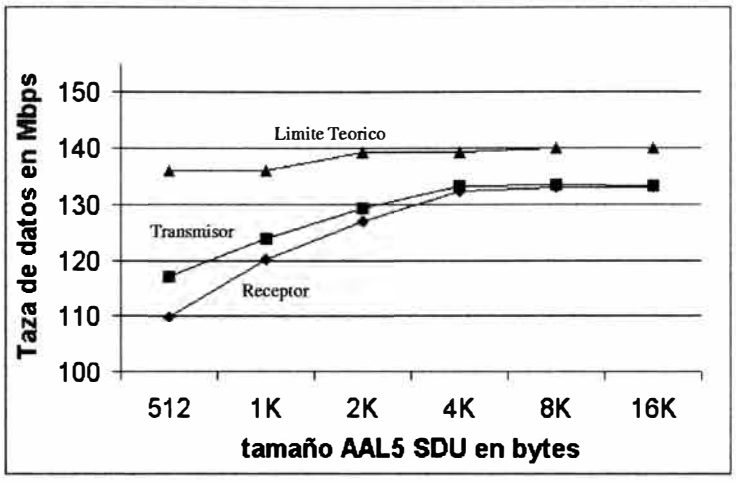

Figura 4.4: Valores experimentales de throughput para distintos tamaños de SDU

conexión en lazo cerrado (loopback) en el terminal ken, empleando la aplicación `ttcp` con los mismos parámetros empleados en el caso anterior.

La figura 4.5 muestra los resultados obtenidos en forma experimental para el caso de configuración en loopback.

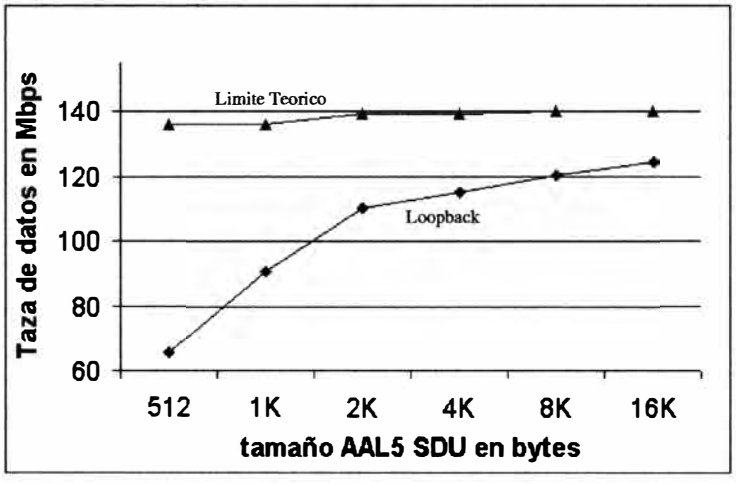

Figura 4.5: Valores experimentales throughput para trasmisiones en loopback de datagramas AAL5 para distintos tamaños SDU

4.4.1 Medida de performance de la facilidad ATM sobre TCP

Como se indicó anteriormente, parte de la plataforma ATM implementada emplea la facilidad de ATM-TCP, como es el caso de la aplicación de audioconferencia

la cual emplea dicha facilidad para poder integrar los clientes de la subred Ethernet, por lo cual también efectuamos el análisis de performance para este caso.

La ecuación 4.2, permite determinar la cantidad de bytes generados a partir de la aplicación hasta alcanzar la capa ATM, para el caso de la aplicación `atmtcp`, la misma que permite emular ATM sobre TCP , las componentes del estándar ATM son implementas por software, luego de esto la información es enviada a través de las subcapas TCP/IP, hasta alcanzar la capa física Ethernet. De esta manera, sea:

$$P = \frac{53\,(F + 8 + (F + 8) mod 48)}{48} bytes \tag{4.4}$$

Donde P corresponde al número de bytes de información o carga de información a ser enviados sobre TCP/IP. Cuando los P bytes alcanzan los niveles del protocolo TCP/IP, se le añade un apéndice de 40 bytes. La carga de información es luego dividida en fragmentos de 1500 bytes en la capa Ethernet. Donde a cada fragmento se adiciona un total de 26 bytes entre apéndice y cabecera, de acuerdo al estandar 802.3. Luego el número total de fragmentos creados será al menos:

$$\frac{(P + 66)\,bytes}{1500 bytes/fragmento} \tag{4.5}$$

El número total de bytes transmitidos es:

$$\frac{1526\,(P + 66)}{1500} bytes \tag{4.6}$$

La capa física en este caso esta representada por la interface Ethernet la misma que ofrece una tasa de bits nominal de 10 Mbps. Así, el máximo troughput estará en el límite de:

$$10\left(\frac{1500P}{1526(P + 66)}\right) Mbps. \tag{4.7}$$

La figura 4.6 representa en forma gráfica los valores nominales de throughput obtenidos para distintos tamaños SDU, para el caso de la facilidad ATM sobre TCP.

Las medidas experimentales de performance ATM sobre TCP, fueron hechas empleando los mismos terminales de las pruebas anteriores, conectados en forma directa por medio de sus interfaces Ethernet, ambos corriendo la aplicación `atmtcp`.

La figura 4.7 muestra el throughput para el caso de transmisiónes unidireccionales de datagramas AAL5 con distintos tamaños de SDU, sobre las respectivas interfaces virtuales ATM-TCP.

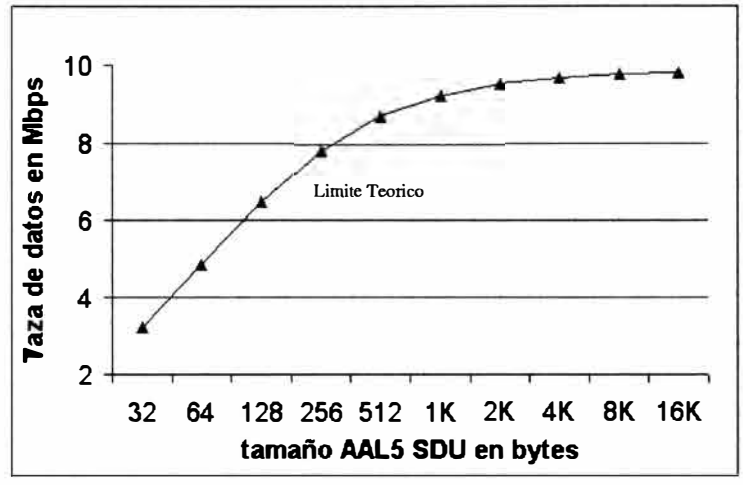

Figura 4.6: Valores teóricos de throughput para distintos tamaños de SDU, caso ATM sobre TCP

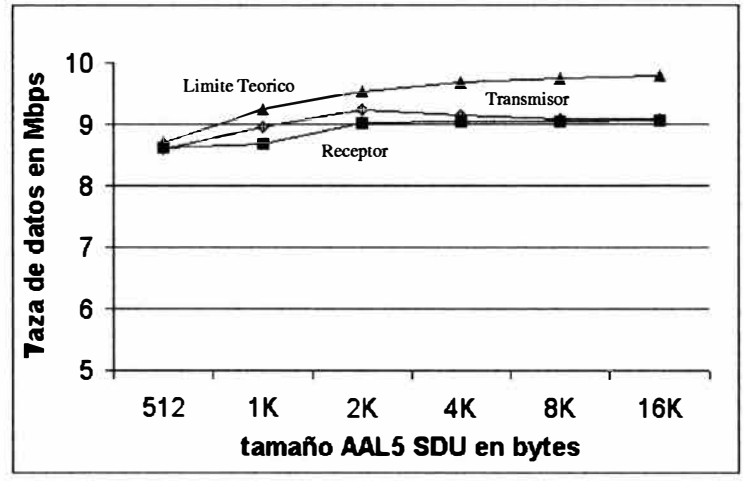

Figura 4.7: Valores experimentales de throughput para distintos tamaños de SDU, caso ATM sobre TCP

Capítulo 5

Aplicación de AudioConferencia

5.1 Conceptos y aspectos complementarios

5.1.1 Multidifusión

Dar una definición que cubra todos los aspectos de multidifusión no es muy fácil, debido a que ésta debe ser lo suficientemente general como para cubrir todas las formas de mutidifusión en un nivel muy abstracto. La siguiente es un intento de definición de multidifusión en general,ver [7]:

Definición de multidifusión : *En una "transmisión"de multidifusión una entidad envía una copia simple de un mensaje a un "proveedor de servicio"(SP) por medio de una operación simple. El SP entrega una copia del mensaje a cada entidad destino de la transmisión de multidifusión.*

El número de entidades destino puede ser 0, todas o cualquier número intermedio. Difusión simple (una entidad destino) y difusión total (Broadcasting) son casos especiales del método de multidifusión. La figura 5.1 muestra una representación gráfica.

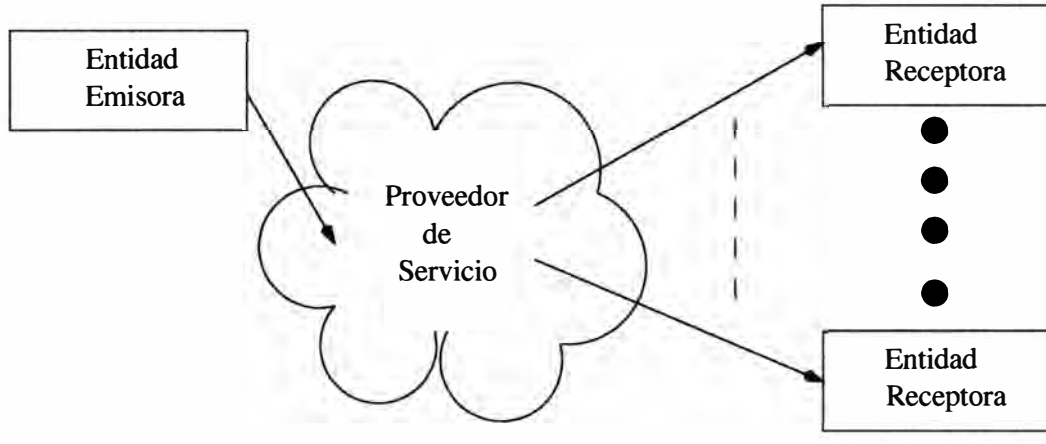

Figura 5.1: Representación de definición de Multidifusión

Discusión

Para el propósito de esta discusión, se asume que existen dos posibles tipos de proveedores de servicio; proveedores de servicio multidifusión y proveedores de servicio unidifusión. Si no se provee un servicio de multidifusión a nivel de red,lo que dependerá de la topología, hay algunas que son adecuadas para este tipo de servicio, una capa alta puede ofrecer dicho servicio empleando un método de réplica y envío en su propio nivel.

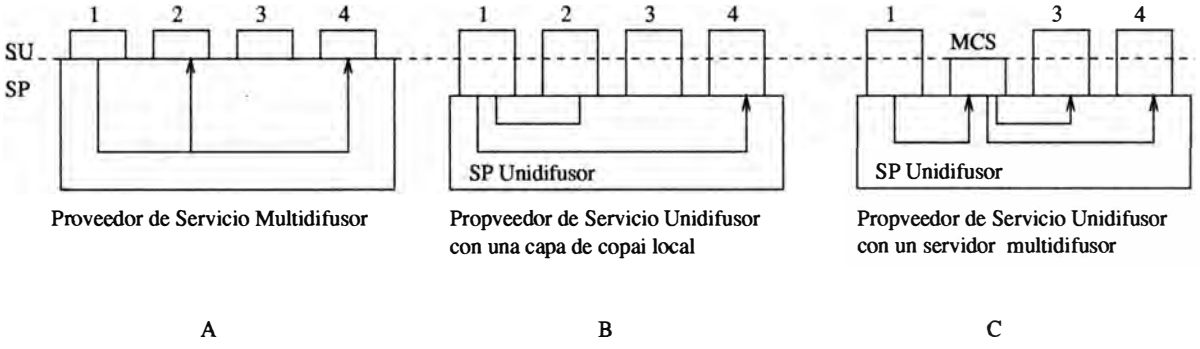

Figura 5.2: Multidifusión en diferentes capas de protocolo.

En la figura 5.2 se muesta un ejemplo de un proveedor de servicio multidifusor y dos representaciones de proveedores de servicio unidifusores con la adición de una capa que provee el servicio de multidifusión. En las figuras 5.2B y C , la combinación de de un SP unidifusor y el servicio de replicación puede ser visto como el servicio multidifusor mostrado en la figura 5.2 A.

En la figura 5.2 B la entidad proveedora del servicio de replicación está localizada en el nodo de envío de información. El servicio multidifusor pasa el paquete de información a la entidad de replicación, la cual luego realiza una réplica para cada receptor y envía ésta a cada uno de ellos usando transmisión por unidifusión.

En la figura 5.2 C, el servicio de réplica está localizado en un nodo conectado a los nodos clientes del servicio de multidifusión. Un cliente del servicio de multidifusión pasa un paquete de datos a ser multidifundido a un servidor proveedor del servicio de multidifusión. Este su vez pasa el paquete a su entidad replicadora localizada en el mismo nodo , luego de lo cual las réplicas son enviadas a cada uno de los nodos destino.

Siendo la multidifusión la base de las aplicaciones de AudioConferencia y debido a que no se dispone actualmente en el desarrollo de la API ATM, es necesario plantear una alternativa de implemententación de este concepto, a fin de poder ofrecer el servicio de audioconferencia a varios nodos.

5.2 Configuración plataforma de aplicación de AudioConferencia

La figura 5.3 muestra la configuración de la plataforma ATM para el desarrollo de la aplicación de audioconferencia.

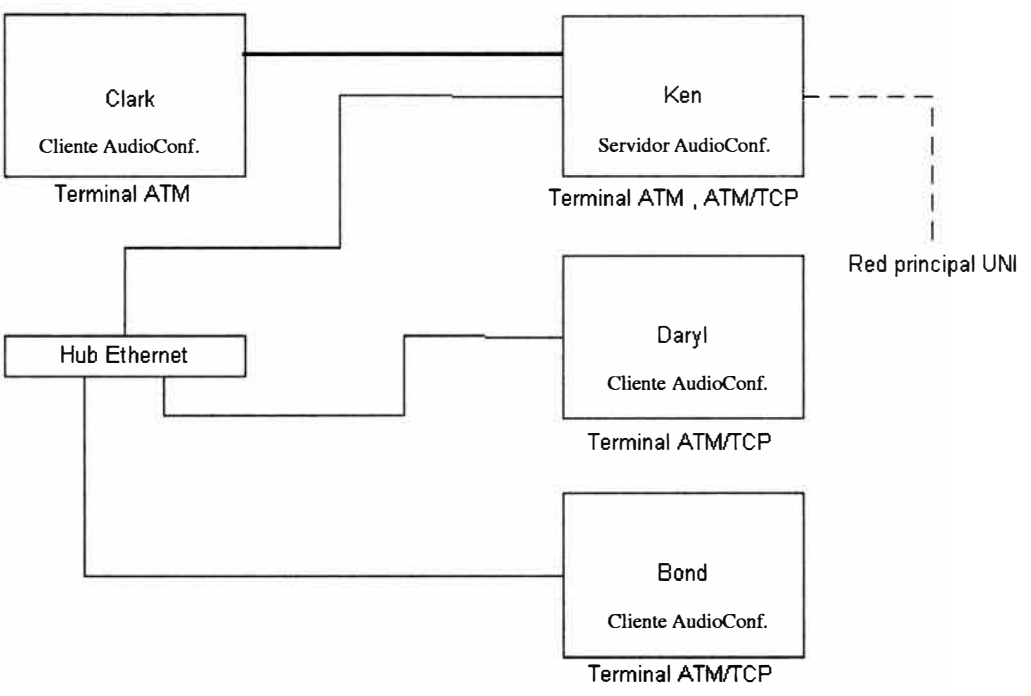

Figura 5.3: Configuración Aplicación Audioconferencia

Como se aprecia en la figura 5.3 el nodo Ken actua como servidor de AudioConferencia, al cual se encuentran conectados los clientes Clark (ATM), Daryl (ATM-TCP), Bond (ATM-TCP), contando con una interface Ethernet adicional para la conexión a la red TCP/IP de la UNI.

5.3 Descripción de la aplicación de AudioConferencia

La aplicación de audioconferencia está organizada en un esquema de enrutamiento centralizado, topología estrella, con un terminal central servidor al cual se encuentran conectados los terminales clientes, una característica clave de este tipo de aplicación es que el servidor es un medio de recepción compartido. A éste se encuentran conectados todos los clientes, y cualquier información transmitida por uno de éstos, está disponible para que los otros clientes conectados puedan acceder a

ella. Si dos clientes transmiten durante el mismo periodo de tiempo, sus señales de información pueden solaparse y distorsionarse. Consiguientemente, sólo un cliente puede transmitir con éxito en un momento dado.

El servidor emplea un esquema de arbitraje centralizado asíncrono en el que cada cliente puede generar un mensaje de petición de transmisión y de fin de transmisión por medio de un PVC de control. El algoritmo de arbitraje utiliza un procedimiento de primero-en llegar-primero en servirse. La figura 5.4 muestra el esquema de arbitraje del servidor junto con las conexiones virtuales de recepción, transmisión y control.

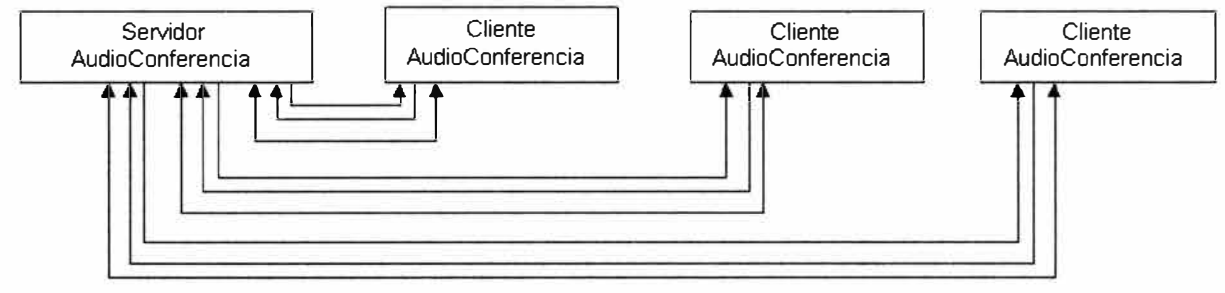

Figura 5.4: Esquema de arbitraje de sevidor de Audioconferencia.

El servidor de audioconferencia maneja un proceso encargado de difundir la información proveniente del cliente maestro en modo transmisión hacia el resto de clientes en modo recepción sin devolver el paquete de información al módulo cliente origen de la transmisión. Adicionalmente maneja el control y arbitraje del sistema por medio de un hilo estándar POSIX, el cual recibe las señales de: petición de transmisión, liberación de canal de transmisión, registro de cliente, reporte de cliente activo; adicionalmente se encarga de actualizar la lista clientes activos de los terminales cliente además de la indicación de cliente maestro de transmisión actual. El servidor muestra también información estadística de tiempo de participación por clientes, a fin de permitir la tarificación por tiempo de uso del sistema. La figura 5.5 muestra el diagrama de bloques de la aplicación servidor de audioconferencia.

Los clientes estan basados en la ejecución de tres hilos estándar POSIX, uno de los cuales se encarga de las funciones de lectura de audio y transmisión del paquete de información directamente sobre un socket ATM AAL5-UBR, otro hilo se encarga de la lectura del socket de recepción ATM AAL5-UBR y su reproducción en el dispositivo de audio correspondiente, y finalmente el tercero se encarga de las funciones de control y coordinación con el módulo servidor de audioconferencia. La figura 5.6 muestra el diagrama de flujo de la aplicación cliente de audioconferencia.

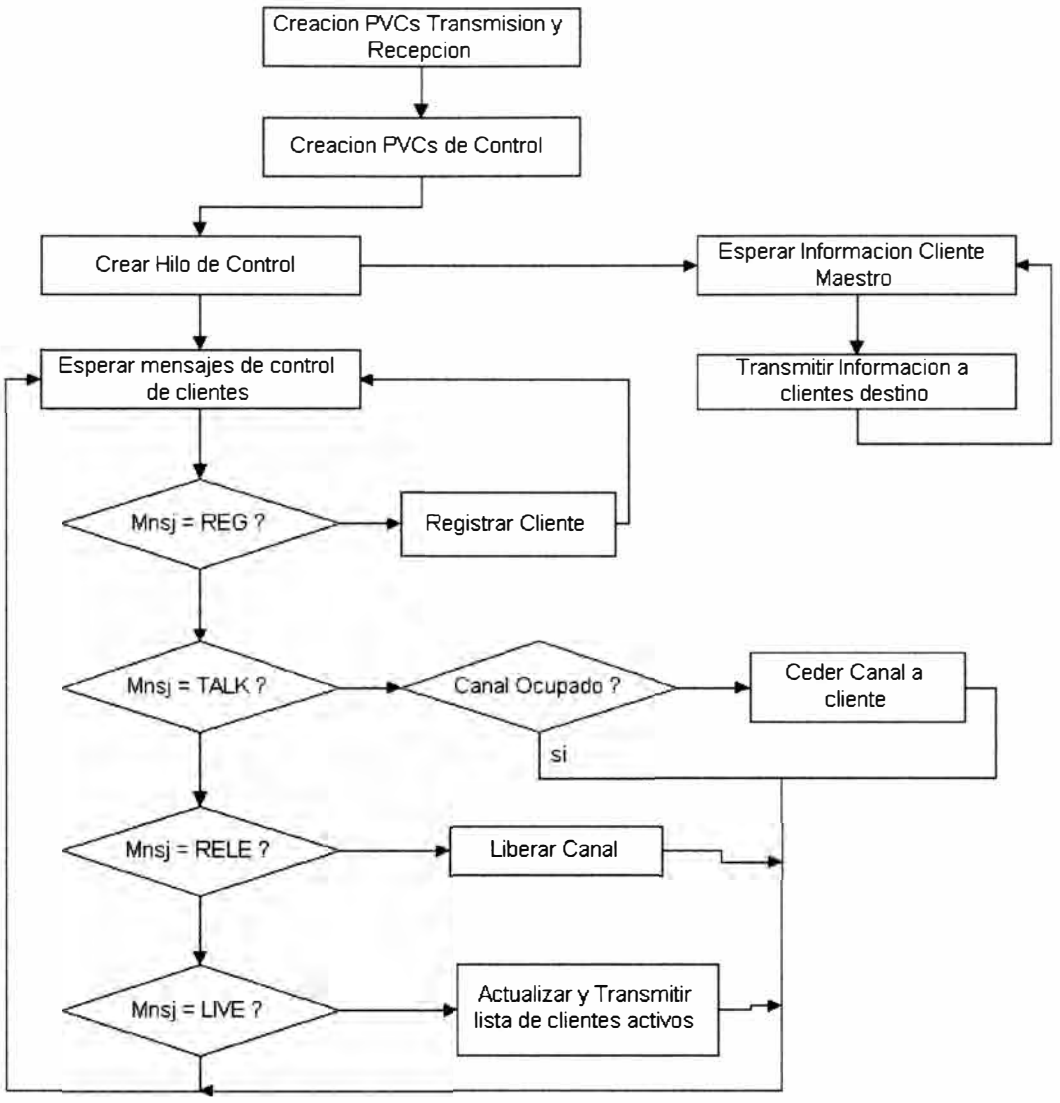

Figura 5.5: Diagrama de flujo de aplicación Servidor

5.4 Programación de los dispositivos de audio en Linux

OSS (Open Sound System) es uno de los manejadores de dispositivo para acceso a tarjetas de sonido y otros dispositivos de audio bajo varios sistemas operativos Unix ,incluyendo Linux. La versión actual soporta la mayoría de tarjetas de sonido y dispositivos de audio comerciales.

Las aplicaciones en Linux no acceden (normalmente) al hardware de audio en forma directa. Todos los datos que estan siendo grabados o reproducidos son almacenados en un buffer DMA del kernel, mientras el dispositivo accesa a éste. La aplicación usa las primitivas `read` y `write` normales para transferir datos entre el

buffer del kernel y el buffer en el segmento de datos de la aplicación.

El manejador del dispositivo de audio, emplea una versión mejorada del llamado método de `doble buffering`. En la base de este método existen dos buffers. Uno de ellos es accedido por el dispositivo mientras que el otro es leído o escrito por la aplicación, al mismo tiempo. Cuando el dispositivo ha procesado el primer buffer, éste se mueve hacia el otro. Este proceso es repetido durante todo el uso del dispositivo. Este método da a la aplicación tiempo para poder hacer algún procesamiento al mismo tiempo que el dispositivo se encuentra corriendo. Esto hace posible grabar y reproducir sin pausas.

La cantidad de tiempo que la aplicación puede tomar en procesar la mitad de buffer depende de el tamaño de buffer y de la taza de datos. Por ejemplo cuando un programa se encuentra grabando audio usando muestreo de 8 kHz / 8 bits / mono, la taza de datos es de 8 kilobytes / segundo. Si el buffer es de 2*4 kbytes, éste da a la aplicación más de 0.5 seg. de tiempo para grabar los datos a disco o para regresar y leer datos del dispositivo. Si la aplicación se toma más de 0.5 segundos, el buffer se sobresatura (overrun) y el manejador de dispositivo tendrá que descartar algo de información. Sin embargo las cosas se vuelven mas complicadas cuando la taza de datos es incrementada. Por ejemplo a nivel de calidad de audio CD donde la taza de datos es de 172 kilobytes/seg., el tiempo disponible es de sólamente 23 milisegundos. Es notable que los resultados pueden ser mejorados usando buffers mas grandes, sin embargo esto incrementaría las latencias referidas a los buffers.

El metodo usado por el manejador de dispositivo de OSS puede ser llamado *multi-buffering.* En este método, el espacio de buffer disponible es dividido en bloques de igual tamaño los cuales son llamados fragmentos. En este sentido es posible incrementar el tamaño del buffer sin incrementar las latencias relacionadas con el buffer. Por defecto el manejador de dispositivo computa los tamaños de los fragmentos de tal modo que las latencias son de 0.5 seg (para salida) o de 0.1 seg. (para entrada). Existe también la posibilidad de ajustar el tamaño de los fragmentos por medio de la primitiva `ioctl` en el caso de aplicaciones que requieran el uso de diferente tamaño.

Las tarjetas de sonido normalmente tienen dispositivos separados o puertos para la producción o grabación de sonido. Exiten diferencias entre las diferentes tarjetas pero la mayoría de ellas contienen los siguientes dispositivos.

- El Dispositivo de voz digitalizada (usualmente llamados codec, PCM, DSP o ADC/DAC) es usado para la grabación y reproducción de voz digitalizada.

- El dispositivo mixer es usado para el contol de niveles de volumen de los diferentes dispositivos de entrada y salida. El mixer también permite controlar la conmutación de la fuente de entrada de sonido ya sea de la línea del micrófono, la línea de entrada de audio y la línea de entrada de CD.

- El dispositivo sintetizador es usado principalmente para tocar música, también para la generación de efectos de audio en aplicaciones de juego.

- La interface MIDI, es un dispositivo usado para conectar sintetizadores externos a la computadora.

La API(interface de programación) del controlador OSS esta definido en la cabecera `sys/soundcard.h`, escrita en lenguaje C. Este archivo de cabecera es distribuído con el controlador, el cual se incluye como parte de la distribución de Linux.

Entre los archivos dispositivo soportados por OSS tenemos

- /dev/mixer

- /dev/sndstat

- /dev/dsp y /dev/audio : Estos son los dispositivos principales para aplicaciones de voz digitalizada. Cualquier dato escrito en este dispositivo es procesado con el dispositivo DAC/DSP/PCM de la tarjeta de sonido. La lectura de este dispositivo retorna datos de audio grabado de la fuente seleccionada de sonido (por defecto la entrada de microfono). Los archivos de dispositivo /dev/audio y /dev/dsp son muy similares. La diferencia es que /dev/audio usa codificación u-Law logarítmica por defecto, mientras que /dev/dsp usa codificación lineal de 8 bits sin signo. Debemos notar que el formato de muestreo inicial es la única diferencia entre estos archivos de dispositivo. Ambos dispositivos tienen un comportamiento similar despues que son programados con sus respectivos parámetros haciendo uso de llamadas ioctl().

- /dev/sequencer

- /dev/music

- /dev/midi

En forma breve, es posible actuar sobre estos archivos de dispositivo usando llamadas al sistema tal como open() , close(), read(), write(). Además es posible también cambiar los parámetros de los dispositivos llamando a la función ioctl() con sus respectivos argumentos. La mayoría de los dispositivos tiene la capacidad de trabajar en modo half duplex lo que significa que pueden grabar y reproducir pero no al mismo tiempo. A los dispositivos que pueden grabar y reproducir al mismo tiempo se les conoce como dispositivos full duplex. Los dispositivos de audio son siempre abiertos en forma exclusiva. Si otro programa trata de abrir un dispositivo

ya abierto , el manejador retorna inmediatamente un error (EBUSY).

El proceso de inicialización del dispositivo sonido dentro de la aplicación cliente, ejecuta las respectivas llamadas al sistema para abrir el dispositivo /dev/audio y programar los siguientes parámetros:

- Número de canales de entrada:1 canal mono.
- Resolución de muestreo: 8 bits.
- Velocidad de muestreo: 22050 Hz.
- Tamaño y número de fragmentos buffer de sonido: la programación de este último parámetro es importante en aplicaciones de sonido en tiempo real, como es el caso de la aplicación de audioconferencia considerada en esta tesis. El API OSS provee de una llamada ioctl para la programación del tamaño de fragmentos y número máximo de fragmentos, la cual esta dada como:

```
int arg = 0xMMMMSSSS;

if (ioctl(audio_fd, SNDCTL_DSP_SETFRAGMENT, &arg)) error();
```

El argumento de esta llamada es un entero codificado como 0xMMMMSSSS (en hexadecimal). Los 16 bits menos significativos determinan el tamaño del fragmento. El tamaño es 2^{SSSS}. Por ejemplo SSSS=0008 da un tamaño de 256 bytes. El tamaño mínimo es de 16 bytes y el tamaño máximo es igual a la mitad del buffer total del dispostivo. Los 16 bits más significativos (MMMM) determinan el número máximo de fragmentos. El valor mínimo es de 2 y el máximo depende de situaciones particulares.

En vista que el hardware de audio del que se dispone es del tipo half-duplex, se ha considerado la programación de la aplicación cliente basada en dos modos de funcionamiento : modo transmisión (hablando) y modo recepción(escuchando).

5.5 Análisis y alcances de la aplicación

El presente análisis se centra en determinar el número límite de clientes de AudioConferencia y los principales parámetros que influyen dentro de la determinación de dicho límite.

Como se expuso anteriormente la aplicación de Audioconferencia desarrollada se encuentra dividida en una aplicación servidor de audioconferencia y una aplicación cliente corriendo en cada terminal que forma parte de una sesión dada de audioconferencia.

La aplicación de audioconferencia, presenta una configuración mixta basada en un solo cliente ATM ,y el resto de clientes Ethernet por medio de la facilidad ATM sobre TCP, conectados al servidor de audioconferencia, presentando un sistema de arbitraje centralizado, tal como se describió anteriormente.

Dentro de la aplicación cliente se configuran los parámetros de programación del dispositivo de audio y de SDU máximo para la transmisión de la información de audio digitalizada, dichos parámetros están fijados en

- Frecuencia de muestreo: 22050 Hz.
- Resolución: 8 bits.
- Número de fragmentos buffer de sonido: 10 fragmentos.
- Tamaño de fragmentos buffer de sonido: 512 bytes.
- Tamaño SDU AAL5: 1024 bytes.

De acuerdo a estos parámetros observamos que la aplicación dispone de 23.22 mseg para poder enviar la información de audio almacenada en un fragmento del buffer de sonido hacia el servidor,sin perdida o sobreescritura de fragmento,lo que afecta la calidad de sonido; el servidor a su vez deberá efectuar la multidifusión de la información recepcionada, multiplexando los clientes ATM-TCP, también dentro de este intervalo; de este modo, de acuerdo a la performance nominal encontrada para transmisiones de paquetes AAL5 con tamaño SDU de 1024 bytes, observamos que en el caso de ATM sobre TCP el ancho de banda disponible es de 9.23 Mbps, luego la capacidad total de información a ser distribuida en este intervalo será:

$$23.22 mseg \frac{9.23}{8} Mbytes/seg = 26790 bytes \tag{5.1}$$

De donde la cantidad clientes de audioconferencia comprendidos dentro de esta carga de información estará determinada por:

$$\frac{26790 bytes}{512 bytes/cliente} = 52 clientes \tag{5.2}$$

Luego la capacidad máxima de clientes de audioconferencia sin degradación de calidad de sonido para los parámetros mostrados será de 52 clientes.

En resumen notamos que los principales parámetros que afectan tanto el número de clientes como la calidad de sonido proporcionada por la aplicación de AudioConferencia son la frecuencia de muestreo del dispositivo de sonido y el tamaño de SDU AAL5.

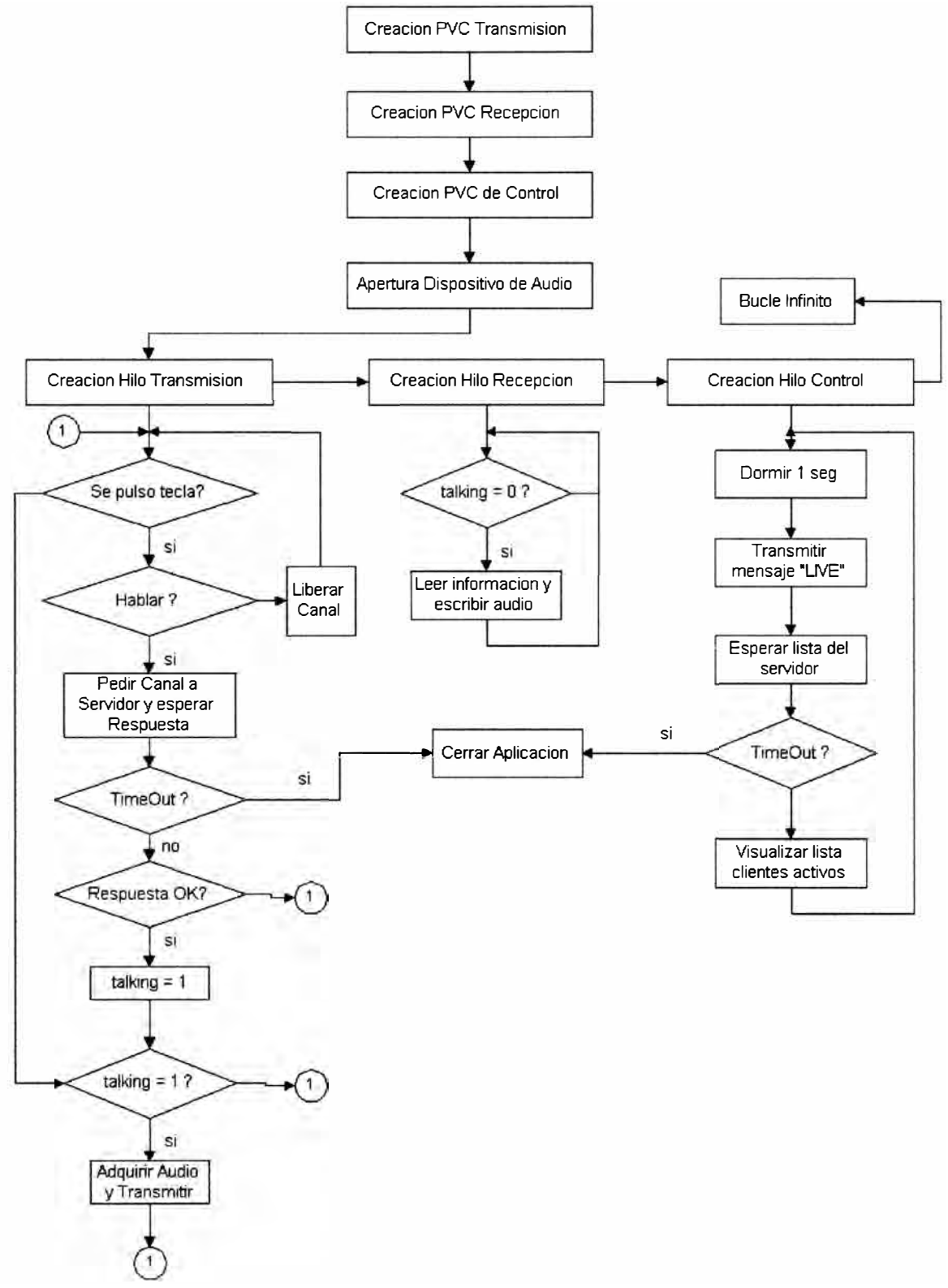

Figura 5.6: Diagrama de flujo de aplicación Cliente

Capítulo 6

Conclusiones

Las conclusiones de este trabajo, basados en los resultados de la plataforma ATM implementada y el desarrollo de la Aplicación de AudioConferencia considerada, se resumen en los siguientes puntos:

1. El sistema operativo Linux ofrece grandes ventajas y facilidades en el desarrollo de aplicaciones concurrentes basadas en la ejecución de hilos POSIX, además de aplicaciones de inter-comunicación de procesos basadas en sockets y aplicaciones de adquisición, tratamiento y reproducción de audio, en tiempo real, en comparación con otros sistemas operativos.

2. Se ha observado que la versión 0.59 de distribución de Linux ATM es ampliamente estable, integrada dentro del kernel 2.2.9, en comparación con otras versiones de distribución.

3. De los resultados de performance de la plataforma ATM implementada para el caso de datagramas AAL5, podemos afirmar que la máxima medida de tasa de transmisión de datos es de 133.0474 Mbps para un SDU de 16 KB, a nivel del enlace experimental.

4. Vemos que es posible implementar a nivel de espacio de usuario soluciones concretas a problemas de aplicación de la tecnología ATM en el campo de redes de terminales finales, como por ejemplo multidifusión.

5. Se puede afirmar que el protocolo ATM AAL5 es una buena alternativa para el desarrollo de aplicaciones basadas en la integración de datos de red y audio a partir de terminales finales multimedia. Tal como lo demuestra el desarrollo de la presente aplicación de audioconferencia, donde es posible establecer una sesión de audioconferencia de 15 terminales, con las siguientes características de audio : velocidad de muestreo de 22050 KHz, resolución de 8 bits, mono.

6. La API ATM Linux basada en sockets BSD ofrece grandes ventajas y beneficios para el rápido y fácil desarrollo de aplicaciones ATM en terminales finales, ya

que es posible aplicar la mayoría de los conceptos de la teoría clásica de sockets BSD, con el beneficio adicional de la tecnología ATM.

7. El método de multi-buffering del manejador de dispositivo de audio es altamente eficiente para el desarrollo de aplicaciones de audio en tiempo real.

8. Dentro de los parámetros de inicialización del dispositivo de Audio, se observó que, el parámetro de programación de tamaño y número de fragmentos de buffer de audio influye dentro del tiempo de retardo de grabación y reproducción de audio, debiendo citar además la velocidad de procesamiento del terminal, por lo que es necesario tener en cuenta un compromiso entre el tamaño de buffer fijado con respecto a la velocidad de procesamiento del terminal, habiendo sido estandarizado en nuestro caso en 10 fragmentos de 512 bytes cada uno, para todos los terminales.

9. El número máximo de clientes de la aplicación es de 15, límite impuesto para el desarrollo de la aplicación de audioconferencia, para una velocidad de muestreo de 22050 Hz a 8 bits de resolución en la adquisición de audio.

6.1 Recomendaciones para trabajos futuros

A partir de la plataforma ATM implementada en la presente tesis, teniendo en cuenta su performance y el desarrollo de la aplicación nativa de audioconferencia, es posible la realización de una amplia variedad de futuros trabajos, como por ejemplo:

1. Integración de la aplicación de audioconferencia ATM con una aplicación de audioconferencia TCP/IP.

2. Desarrollo de aplicaciones usando nivel de conexiones SVC.

3. Estudio y evaluación del switch virtual ATM para diferentes protocolos AAL, políticas y esquemas de conmutación de células.

4. Desarrollo de aplicaciones de audioconferencia y videoconferencia dentro de la plataforma integrada de la Universidad, a fin de intercomunicar las distintas facultades y pabellones administrativos, aprovechando los recursos ofrecidos por la tecnología ATM en beneficio de la institución.

5. Integración a nivel metropolitano, nacional e internacional con otros Campus a nivel de tecnología ATM.