Departament d'Arquitectura de Computadors

TESIS DOCTORAL

PROTOCOLO ACTIVO PARA TRANSMISIONES GARANTIZADAS SOBRE UNA ARQUITECTURA DISTRIBUIDA Y MULTIAGENTE EN REDES ATM

Doctorando: José Luis González Sánchez
Director de Tesis: Jordi Domingo Pascual

Mayo de 2.001

SINOPSIS

En esta tesis doctoral se presenta TAP (*Trusted and Active Protocol PDU transfer*), una arquitectura para redes de tecnología ATM, novedosa por sus características distribuida, activa y multiagente. El protocolo propuesto para la arquitectura ofrece transferencias garantizadas a un conjunto privilegiado de conexiones VPI/VCI. Se propone también una extensión de la capa AAL-5 de ATM que hemos denominado EAAL-5 (*Extended AAL type 5*) usada para la gestión de las conexiones privilegiadas extremo-extremo.

TAP ofrece garantía de servicio (GoS) cuando la red está perdiendo células ATM y aprovecha los periodos de inactividad en los enlaces para realizar las retransmisiones de las CPCS-PDU-EAAL-5. El protocolo propuesto emplea mecanismos NACK (mediante células RM de retorno) y es soportado por conmutadores ATM activos equipados con una memoria de almacenamiento de PDU denominada DMTE (*Dynamic Memory to store Trusted native EAAL-5 PDU*). La arquitectura activa propuesta está basada en un SMA (Sistema Multiagente) constituido por agentes programables colaborativos y distribuidos en la red. Las simulaciones realizadas demuestran la efectividad del mecanismo de recuperación de PDU propuesto con un mejor *goodput* en la red.

La arquitectura TAP es soportada sobre conmutadores ATM activos que denominamos AcTMs (*Active Asynchronous Transfer Mode Switch*) y que hemos diseñado con técnicas software para: garantizar la gestión justa de colas de entrada basadas en WFQ (*Weighted Fair Queueing*); realizar el control de congestiones del buffer inspirado en EPD (*Early Packet Discard*); y evitar, con *VC Merge*, la mezcla de las PDU de conexiones diferentes. Estas técnicas software se proponen, por tanto, con la intención de: distribuir de forma justa la carga de los conmutadores; optimizar las retransmisiones de PDU; aliviar la implosión sobre las fuentes; evitar la fragmentación de las PDU y disminuir el *interleaving* de células, optimizando el *goodput*.

Los conmutadores AcTMs requieren también el hardware apropiado para soportar TAP. Para ello, además del buffer, se proponen, la memoria DMTE y un conjunto de tablas de E/S asociadas a cada uno de los puertos de los AcTMs. Se demuestra que estos requerimientos hardware son realistas y viables para ser incorporados en los conmutadores activos. Destacamos el carácter multidisciplinar de esta tesis, donde la base de las investigaciones es la ingeniería de protocolos ATM, complementada con las novedosas ventajas que los agentes software pueden aportar. No obstante, los conmutadores finalmente obtenidos podrían ser objeto del ámbito de las arquitecturas especializadas, de forma que varios módulos del prototipo presentado, podrían ser implementados como componentes hardware para optimizar su rendimiento.

Una vez identificadas las limitaciones de la tecnología ATM para soportar las transferencias garantizadas, que son nuestro principal objetivo, se describe la motivación general de estas investigaciones en entornos donde ATM es la base del tráfico IP. De este modo, se emplea NS (*Network Simulator*) para el estudio de escenarios donde el protocolo TAP puede aportar importantes beneficios al conocido protocolo TCP.

Para poder estudiar el comportamiento de todas estas propuestas hemos implementado un simulador de TAP que aprovecha las ventajas que aporta el lenguaje Java para el desarrollo de protocolos de comunicaciones y de SMA. Este simulador permite definir múltiples escenarios y analizar los resultados de la simulación del prototipo para poder llegar a una serie de interesantes conclusiones. Las simulaciones a través de fuentes ON/OFF analizan conexiones punto-a-punto y punto-a-multipunto usando clases, objetos, *threads*, sincronizaciones y procesos distribuidos implementados en lenguaje Java.

Esta memoria de tesis doctoral ha sido organizada en tres grandes apartados con el objeto de estructurar adecuadamente los contenidos presentados. La *Parte I* está dedicada a analizar las investigaciones relacionadas con este trabajo, de forma que se describen en siete capítulos los aspectos básicos de la tecnología ATM y se aprovecha cada uno de los capítulos para presentar resumidamente nuestras aportaciones, las cuales son ampliadas en las *Partes II* y *III*. De este modo, comenzamos destacando en el *Capítulo 1* los fundamentos de la tecnología, para pasar después a describir en el *Capítulo 2* una taxonomía de arquitecturas y protocolos para las redes ATM que nos sirven para identificar la propuesta TAP cuya arquitectura básica es incluida al final del capítulo. El *Capítulo 3* se centra en los conceptos de fiabilidad y garantía de servicio (GoS) destacando éste último, ya que es una de nuestras propuestas a los parámetros generales de calidad de servicio (QoS) y que se deriva de éstos. Así, se explica el mecanismo con el que se ofrece la GoS a las fuentes privilegiadas. Seguidamente, el *Capítulo 4* se centra en el control de congestión y la justicia, ambos aplicados sobre las colas de entrada de los conmutadores ATM. Éstos son también dos aspectos básicos en nuestra propuesta para conseguir aportar soluciones al problema de las congestiones en las fuentes privilegiadas, pero garantizando además la justicia a aquellas fuentes que no lo son. Una vez estudiadas las propuestas de la literatura se presenta un esquema de nuestro algoritmo QPWFQ. El *Capítulo 5* estudia los diversos mecanismos de control de congestión aplicados sobre el buffer de los conmutadores y, después de analizar las propuestas más extendidas, comentamos nuestro algoritmo EPDR inspirado en EPD para conseguir atender las solicitudes de retransmisión de las PDU congestionadas. En el *Capítulo 6* se realiza una revisión de la literatura en materia de agentes software orientada hacia las redes de comunicaciones, con la intención de centrar adecuadamente el SMA que proponemos como soporte de TAP y con el objetivo de conseguir una red activa formada por conmutadores AcTMs cuya arquitectura es adelantada al final de este capítulo. El *Capítulo 7* justifica el carácter distribuido del protocolo TAP sobre una VPN (*Virtual Private Network*) constituida por nodos AcTMs que coexisten con conmutadores no activos en la misma red. En resumen, la *Parte I* trata de justificar nuestras propuestas, reafirmándolas sobre los propios fundamentos de la tecnología actual.

La *Parte II* identifica las motivaciones generales de esta tesis, partiendo de las limitaciones actuales de la tecnología ATM que se pretenden solventar con la propuesta de TAP. Esta parte se ha dividido en dos capítulos, dedicándose el *Capítulo 8* a describir las motivaciones generales, de modo que el control de congestión en los nodos de la red no sólo beneficia al tráfico ATM nativo, sino que puede ser también de utilidad para protocolos tan extendidos como TCP. Se identifican, por tanto, los beneficios aportados por TAP a las redes actuales. En el *Capítulo 9* se discuten las limitaciones de ATM frente al parámetro de GoS propuesto y se explica cómo TAP puede evitar problemas tan indeseables como la fragmentación de las PDU, el *interleaving* del tráfico, las retransmisiones extremo-extremo y la implosión en las fuentes de tráfico.

El objetivo de la *Parte III* es detallar las soluciones propuestas, de forma que en cuatro capítulos se realiza una descripción detallada, tanto de la arquitectura, como del protocolo que se implementa sobre ella. El *Capítulo 10* describe la arquitectura distribuida y multiagente TAP, relacionándola con el modelo arquitectónico ATM, y analizando cada uno de los componentes hardware y software de los conmutadores AcTMs. El *Capítulo 11* se centra específicamente en detallar el conjunto de algoritmos que constituyen el protocolo TAP y, por tanto, en el SMA que lo constituye. También se formaliza la idea intuitiva de aprovechar los tiempos de inactividad de la red para atender las retransmisiones de las PDU congestionadas. El *Capítulo 12* presenta los detalles de implementación del simulador de TAP que proponemos como prototipo para analizar los resultados obtenidos en diversos escenarios. Se argumenta la elección del lenguaje Java como herramienta para el desarrollo de protocolos y SMA, para pasar después a describir la metodología y las decisiones de diseño más importantes, así como de las clases Java más destacables del prototipo. Este capítulo concluye con el análisis de los resultados más significativos de las simulaciones. Por último, el *Capítulo 13* se dedica a identificar líneas futuras de acción que aporten continuidad al conjunto de investigaciones de las que ha sido objeto esta tesis doctoral.

Antes de concluir deseamos dejar testimonio del más sincero agradecimiento al Dr. Jordi Domingo Pascual que, con su experiencia y profundos conocimientos técnicos, ha colaborado activamente a que esta tesis llegue a su conclusión y que, con su extraordinaria calidad humana, ha logrado mantener viva la colaboración durante cinco años y a mil kilómetros de distancia.

Cáceres, Mayo de 2.001 José Luis González Sánchez

ÍNDICE

CAPÍTULO 1

FUNDAMENTOS DE LA TECNOLOGÍA ATM

1.1. INTRODUCCIÓN

El Modo de Transferencia Asíncrono (ATM) es la base de un estándar internacional propuesto como un sistema de conmutación de paquetes para el transporte de datos basado en células de longitud fija y de pequeño tamaño (53 octetos). Se caracteriza también ATM por ofrecer funciones de gestión de tráfico para la transferencia óptima de información en tiempo real (tráfico multimedia) y tiempo no-real (datos o imágenes estáticas), aportando la atractiva ventaja de la integración de diferentes flujos de información.

La tecnología ATM recibe su máximo impulso cuando es elegida por CCITT como la base para la implementación de la RDSI-BA (Red Digital de Servicios Integrados de Banda Ancha) [1,2]. Desde ese momento, y a lo largo de más de una década, la tecnología ha experimentado una continua evolución siendo fuente de constantes investigaciones que han permitido que sea en la actualidad la solución más sólida en las troncales internacionales de largo alcance. Otra de las ventajas de ATM es precisamente su escalabilidad lo que le permite ser implantada tanto en entornos WAN como MAN y LAN.

La integración de servicios y la escalabilidad, ligadas al extraordinario rendimiento de los medios de transmisión ópticos que soportan velocidades de transferencia del orden de Gbps, con probabilidades de error en torno a 10^{-12}, han permitido que ATM de respuesta a las actuales necesidades de comunicación a través de aplicaciones de toda índole.

La característica orientada a la conexión de ATM es uno más de los aspectos diferenciadores con otras tecnologías claramente implantadas como las redes IP. El hecho de ser una tecnología orientada a conexión hace que las transferencias se realicen a través de circuitos virtuales (VPI/VCI) establecidos extremo-a-extremo, y los cuáles se mantienen abiertos durante todo el tiempo que dura la comunicación. Estos circuitos virtuales son creados en la fase de establecimiento de la conexión que es cuando el usuario de la red puede especificar los parámetros de tráfico que va a generar, o los recursos de red que va a requerir. Así, el usuario negocia la calidad del servicio que espera recibir, de forma que la propia red dispone de mecanismos de gestión de recursos como la función CAC (Control de Admisión de la Comunicación), y como la función UPC (Control de Parámetros de Uso). La función CAC, usada para la negociación de la conexión, actúa a modo de control de flujo impidiendo la entrada de usuarios para los que la red no dispone de recursos, y evitar la sobrecarga de ésta. La función UPC se encarga de velar, durante la comunicación, por el buen cumplimiento del contrato de tráfico que los usuarios han negociado con la función CAC en el establecimiento de la conexión. Vemos como ATM no sólo se encarga de atender las necesidades de conexiones concretas, sino también del estado general de la misma, de forma que se intentan evitar situaciones de sobredimensión que conduzcan a congestiones o sobrecargas.

A la vista de esta descripción general de las características más importantes de la tecnología podemos adivinar que, aunque muchos aspectos de la misma están resueltos, aún quedan muchos otros que son foco de atención de la comunidad investigadora que realiza propuestas para mejorar la tecnología. Así, para centrar las motivaciones generales de esta tesis deseamos realizar en este primer capítulo una descripción de los aspectos de ATM que están más directamente relacionados con nuestras investigaciones.

Ya que aspiramos a movernos en el ámbito de las arquitecturas, protocolos y tráfico nativo ATM, requerimos conocer, tanto el formato de las células ATM, como el modelo arquitectónico para identificar dónde realizaremos nuestras aportaciones, las cuáles serán descritas detalladamente en capítulos posteriores.

Como uno de nuestros objetivos es la garantía de servicio a las transferencias, vamos a centrar los aspectos relacionados con las clases de servicio, así como los parámetros de calidad de servicio para acabar llegando al mapeo de las primeras con los segundos, todo ello desde el punto de vista de las recomendaciones de los estándares.

Nuestra motivación general es la de dar solución a las situaciones de congestión en los conmutadores ATM, por lo que vamos a ver cómo la tecnología ofrece diversos mecanismos de control de errores, de control de flujo y de control de congestión, pero todos ellos realizados extremo-extremo, debido a la característica orientada a la conexión. Esto acaba perjudicando al rendimiento de la red o al de los propios extremos de la comunicación. Las situaciones de congestión en los conmutadores son impredecibles y cuando aparecen pueden acabar afectando negativamente a toda la red.

1.2. CARACTERÍSTICAS PRINCIPALES DE LA TECNOLOGÍA ATM

La célula constituye la unidad de transferencia de las redes ATM que, como hemos dicho antes, son de tamaño fijo y relativamente pequeñas si se compara con otras tecnologías. De los 53 octetos totales, 5 bytes se emplean para las cabeceras y los 48 restantes se emplean para la carga útil de datos. La *Figura 1.1* muestra cada uno de los campos de las cabeceras con sus tamaños en bits. Se puede observar separadamente el formato de las células UNI (User Network Interface) usadas entre extremos de la comunicación y la red, y el de las células NNI (Network to Network Interface) usadas entre los conmutadores de la red.

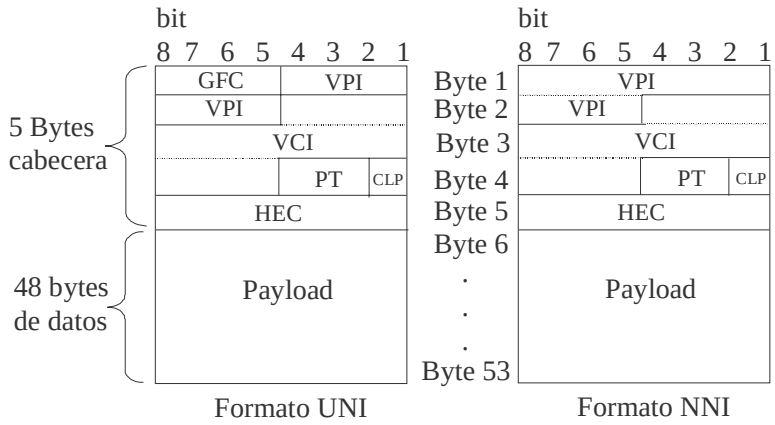

Figura 1.1. *Formato de células ATM*

Cada célula lleva en su cabecera un identificador de camino virtual (VPI) de 8 bits y un identificador de canal virtual (VCI) de 16 bits. El campo PT (Payload Type) de 3 bits se usa para indicar el tipo de información contenida en el campo de carga de datos de la célula. El bit CLP (Cell Loss Priority) lo emplea el emisor para especificar la prioridad deseada cuando aparecen situaciones de congestión. Las células con CLP = 1 tienen más baja prioridad y serán las primeras en ser descartadas por los conmutadores cuando se congestionan. El campo HEC (Header Error Control) se usa como mecanismo de detección de los errores producidos en las transmisiones de las cabeceras. Con los 8 bits del campo HEC se detectan todos los errores de los 32 bits restantes de la cabecera, aportando un mecanismo realmente poderoso ligado a las bajas tasas de error de la fibra óptica que hemos comentado antes.

Una cuestión básica para nosotros es la comprensión del modelo arquitectónico ATM ya que sobre éste vamos a realizar aportaciones con la doble intención de mantenerlo para respetar las características nativas del modelo en nuestra arquitectura, y a la vez de modificarlo para conseguir nuestros objetivos de mejora de la tecnología. Describimos por esto el modelo presentado en la *Figura 1.2* donde podemos observar las tres capas principales de la pila de protocolos que constituyen la arquitectura tridimensional ATM.

El modelo ATM se apoya en las capas Física, ATM y de Adaptación ATM, cada una de ellas con una serie de funciones especificadas para permitir las transferencias de las células. En nuestros objetivos la capa Física permanecerá intacta y nos centraremos principalmente en la capa de adaptación ATM (AAL).

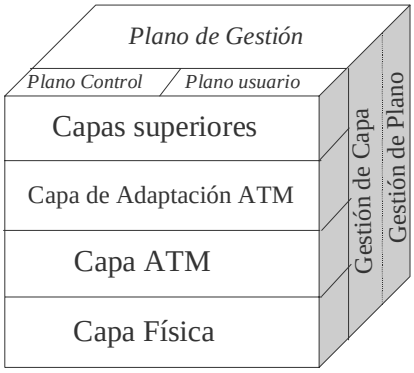

Figura 1.2. *Modelo Arquitectónico ATM*

Algo esencial para el servicio ofrecido por las redes ATM es la capa de Adaptación ATM que permite adaptar los flujos de la capa Física (basados en células) a paquetes, datagramas o flujos de bits propios de las capas superiores de la pila de protocolos. Destacamos especialmente esta capa porque, como veremos en capítulos posteriores, propondremos una extensión de la capa de AAL tipo 5 para conseguir nuestros objetivos. La capa AAL está formada por dos subcapas: en la parte superior la Subcapa de Convergencia (SC) y por debajo de ésta la Subcapa de Segmentación y Reensamblado (SAR). Las funciones de estas capas son las siguientes:

- Subcapa de Convergencia: Realiza la adaptación a la velocidad de transferencia del usuario, la corrección de errores y mantiene la sincronización extremo-extremo y se encarga del control de flujo.
- Subcapa SAR: en el caso del emisor, es la responsable de segmentar el tráfico continuo de tramas de información a células de 48 octetos que son pasadas a la capa ATM inferior. También detecta posibles células erróneas y/o perdidas. En el caso del receptor esta misma subcapa se encarga del reensamblado de las células que recibe desde la capa ATM para convertirlas en PDUs o tramas que van a ser pasadas a los protocolos de las capas superiores.

Una visión complementaria del modelo ATM de la *Figura 1.2* puede ser obtenida observando la *Figura 1.3* donde se representan las capas de la arquitectura, tanto en los nodos de la red como en los nodos locales y en los de usuario. Podemos ver cómo los nodos de la red o conmutadores no disponen de capa de adaptación AAL, ya que el tráfico en la red es completamente nativo ATM y lo que se transmite entre conmutadores son células. Los nodos o conmutadores locales sí soportan la capa AAL, ya que necesitan ajustar el tráfico generado desde los nodos terminales de usuario que, como podemos observar, pueden hacerlo a través de cualquier protocolo de comunicación como TCP/IP. En realidad, las células no son visibles por los usuarios que lo que hacen es generar las tramas propias del protocolo que están usando. Estas tramas son las que AAL se encargará de segmentar en células que pasan a la capa ATM, que no es otra cosa que un modo de transporte de células que son transferidas a la capa Física. La *Figura 1.3* representa también las tres primeras capas del modelo de referencia OSI aportando una comparativa entre los dos modelos arquitectónicos, aunque en este aspecto no existe demasiado consenso sobre la relación entre las capas de cada uno de los modelos.

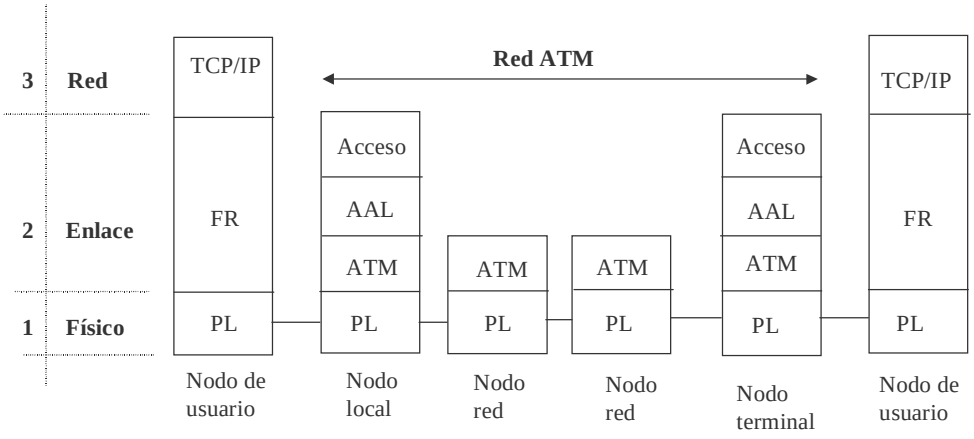

Figura 1.3. *Modelo arquitectónico ATM y RM-OSI*

1.3. CLASES DE SERVICIO ATM

La tecnología *ATM* fue diseñada y desarrollada para soportar e integrar varias clases de servicio como los datos y la información multimedia, generada o no, en tiempo real.

Una conexión es establecida como resultado de la negociación entre el usuario y la red donde se estipula la Clase de Servicio (CoS) a adoptar. La clase de servicio es definida por los parámetros de tráfico de la conexión y por sus parámetros de Calidad de Servicio (QoS). La ITU-T ha propuesto [3,4] varias CoS, definidas como ATC (ATM Transfer Capability), en función de las capacidades de transferencia de información ATM. Cada ATC especifica un conjunto de parámetros y procedimientos de capa ATM para sustentar un modelo de servicio y un conjunto de valores de QoS asociados. Cada ATC es especificada en términos de un modelo de servicio, un descriptor de tráfico, unos procedimientos específicos, una definición de conformidad y los compromisos de QoS demandados por cada conexión, los cuáles la red se compromete a cumplir. Así, ITU-T propone las siguientes ATC:

- DBR (Deterministic Bit Rate). La velocidad de conexión en la fuente es constante y la tolerancia CDV (Cell Delay Variation) es reducida. Esta clase de servicio se emplea en aplicaciones de tiempo real, voz, audio y vídeo. La capacidad de transferencia determinística está pensada para tráfico CBR (Constant Bit Rate) y, por tanto, para responder a compromisos de QoS en términos de pérdidas de células, retardo de transferencia de células y variación del retardo adaptados a CBR. De todos modos, DBR no se limita exclusivamente a aplicaciones CBR y puede usarse en combinación con requisitos de QoS menos rigurosos.

- SBR (Statistic Bit Rate). En este caso la velocidad de conexión es variable y hasta esporádica, y la tolerancia CDV puede ser elevada. La capacidad de transferencia de velocidad binaria estadística usa velocidad de células sostenible y la tolerancia propia de las ráfagas y también la velocidad de células de cresta. SBR es adecuada para aplicaciones en las que existe un conocimiento anterior a las características de tráfico. La QoS se expresa en términos de tasas de pérdidas.

- ABR (Available Bit Rate). Es una clase de servicio especialmente adaptada a los servicios de datos. La velocidad instantánea varía entre un valor máximo PCR (Peak Cell Rate) y un mínimo MCR (Minimum Cell Rate). Cuando la red no dispone de los recursos necesarios se activa un control de flujo para reducir la velocidad de la fuente pero sin caer por debajo del valor MCR que debe estar garantizado. Esta capacidad de transferencia de velocidad binaria disponible está pensada para aplicaciones elásticas en tiempo no real que sean capaces de adaptarse a la anchura de banda instantánea disponible en la red. ABR emplea parámetros estáticos declarados al establecerse la conexión, y parámetros dinámicos renegociables mediante procedimientos de gestión de recursos basados en células RM (Resource Management). El usuario interroga regularmente a la red sobre la anchura de banda disponible en ese momento enviando células RM que transmitan a la red la velocidad solicitada. Existen dos modos de funcionamiento, el de velocidad de células explícita y el modo binario.

- ABT (ATM Block Transfer). Transferencia de bloques ATM es una capacidad destinada a las aplicaciones que pueden adaptar bloque a bloque su velocidad pico de células. Un bloque ATM es un grupo de células delimitado por células RM. ABT emplea parámetros estáticos declarados al establecer la conexión, y parámetros dinámicos renegociables en bloques ATM mediante procedimientos de gestión de recursos usando células RM.

  Existen dos variantes de transferencia ABT. En ABT/DT (Transmisión retardada), la fuente sólo empezará a transmitir un bloque ATM cuando haya recibido un acuse de recibo positivo de la red a través de una célula RM. En la versión de transmisión inmediata, ABT/IT, la fuente inicia la transmisión de las células de usuario después de la célula RM de petición, y la transferencia del bloque ATM sólo se realiza si la red dispone de los recursos solicitados para ese bloque. Si no dispone de recursos, el bloque será descartado. En las dos versiones de ABT la petición de BCR (Block Cell Rate) puede ser elástica, por lo que la red puede decidir seleccionar una BCR menor que la solicitada por la fuente.

- A las cuatro anteriores, y ya clásicas ATC normalizadas en la Rec. I.371 [4], se ha añadido una nueva propuesta como CCT (Controlled Cell Transfer) [5], denominada después CT (Controlled Transfer) [6], desarrollada para soportar servicios de LAN. CT aporta la posibilidad de que todas las ventajas de la red ATM, como la QoS y la integración de aplicaciones puedan ser extractadas. Esta clase de servicio se apoya en un mecanismo de control de flujo basado en créditos y en el empleo de VP y VC. Para el nivel VP el control de flujo se realiza por medio de un clásico mecanismo de ventana variable.

En el nivel VC el control de flujo empleado es el de mensajes BECN (Backward Explicit Congestion Notification). En ambos casos de control de flujo se emplean las células RM para ofrecer realimentación con el emisor. Deben especificarse las direcciones de flujo ya que un nodo puede actuar como emisor y como receptor. La dirección de control de las células RM va siempre desde el receptor al emisor en el caso punto-punto y, en el caso punto-multipunto, la dirección va desde las hojas hasta la raíz. Como no todos los nodos pueden soportar CT, se permite el uso de túneles DBR para unir los puntos en que CT no esté soportado.

Es necesaria la existencia de métodos y capacidades por las cuáles el tráfico de MAN y LAN pueda ser llevado eficientemente a través de redes ATM evitando la formación de islas. Una forma de conseguir esto es disponer de una capacidad de transferencia que sea capaz de ofrecer un servicio de LAN sobre área extensa. Para ofrecer servicios tipo LAN deben cubrirse los requerimientos de LAN habituales como que éstas operan sobre enlaces dedicados de baja latencia y con transferencias de mensajes de longitud variable. La comunicación entre nodos de una LAN es a ráfagas y se producen pocos errores por los mecanismos de control empleados en los protocolos, por lo que se esperan pérdidas mínimas. Pues bien, trasladar todos estos requerimientos en ATM sobre WAN implica la existencia de la capacidad de soportar conexiones permanentes y semipermanentes, manipular las ráfagas de datos amigablemente y ofrecer soporte para entrega de mensajes de longitud variable con bajo retardo. El retardo extremo-extremo en WAN limita la efectividad de las retransmisiones para corregir las pérdidas.

Una aplicación importante para CT es el soporte de conexiones multipunto en LANs virtuales. Para esto lo que se requiere es la posibilidad de que la red ofrezca ancho de banda dinámicamente compartido y la conexión deberá tener garantía en el ancho de banda que se le asigne.

SBR y DBR no pueden compartir ancho de banda eficientemente, ya que si se está en una conexión multipunto-multipunto (mp-mp) no se puede controlar el acceso a la conexión por parte de clientes individuales. DBR y SBR sólo permiten asignar ciertas ratios por cada acceso de cliente, pero si sólo transmiten unos pocos clientes, la conexión puede estar siendo usada a medias. Por otro lado, ABR no garantiza el ancho de banda y realiza, además, un excesivo control. CT sin embargo, permite compartir eficientemente los recursos de red y puede garantizar el ancho de banda. En [5] se demuestra, además, que CT se comporta mejor que DBR y ABT en términos de los retardos experimentados cuando los tres servicios se estudian en condiciones similares.

- La ITU-T, propone en [6] una sexta ATC llamada GFR (Guaranteed Frame Rate) que es también estandarizada por el ATM Forum. GFR, como CT, surge porque bastantes aplicaciones no se caracterizan adecuadamente con las ATC existentes. Esto fuerza a que bastantes de estas aplicaciones sean caracterizadas por las clases de servicios DBR, SBR, ABR o ABT, cuando tienen sus propios parámetros de tráfico. Por ejemplo, la notificación de congestión TCP es implícita y basada en el descenso de paquetes, lo que no puede realizarse mediante la realimentación explícita propuesta en ABR. Otro ejemplo lo podemos encontrar cuando los extremos de la conexión ATM no coinciden con los extremos de la aplicación que está usando la red ATM, es decir, routers que interconectan LANs sobre ATM. Ante esta situación se propone una nueva ATC para aplicaciones que necesitan organizar las células en tramas que son delineadas en la capa AAL. Algunas de estas aplicaciones pueden beneficiarse de una mínima velocidad de tramas garantizada. GFR está pensada para soportar aplicaciones de tiempo no-real con parámetros de tráfico como PCR (Peak Cell Rate), MCR (Minimum Cell Rate), MBS (Maximum Burst Size) y MFS (Maximum Frame Size). La red puede tirar tramas y la QoS no se especifíca, aunque existen dos variantes de GFR:

  - ✓ GFR1, donde la red transporta el bit CLP de forma transparente para las tramas, lo que consiste en que todas las células llevan los bits CLP = 1 ó CLP = 0. La red no puede aplicar etiquetado a las tramas.
  - ✓ GFR2, donde la red puede etiquetar las tramas marcando todas las células de una trama que no pueden pasar el test del algoritmo F-GCRA (Frame-based Generic Cell Rate Algorythm). Esto indica que el bit CLP viaja por la red de forma no transparente y puede ser usado en las situaciones de congestión para rechazar tramas.

Destacamos que, además de las ATC comentadas, la recomendación I.362 especifica una serie de 4 clases de servicio (A, B, C y D) generales relacionadas con servicios concretos. La *Tabla 1.1* muestra algunas de las características de estas CoS.

TABLA 1.1
CLASES DE SERVICIO, SERVICIOS Y CARACTERÍSTICAS

	Clase A	**Clase B**	**Clase C**	**Clase D**
Ejemplo de servicio	Emulación de circuitos, voz y video a velocidad constante	Velocidad variable, voz y video comprimido	Servicios con conexión, transferencia de datos	Datagramas, transferencia de datos
Relación con usuarios	Sincronizada	Sincronizada	No Sincronizada	No sincronizada
Velocidad de acceso	Constante	Variable	Variable	Variable
Modo de conexión	Orientada a conexión	Orientada a conexión	Orientada a conexión	Sin conexión

Además de las clases de servicio recomendadas por la ITU-T, el ATM Forum especifica en [7] un conjunto de categorías de servicio y, para cada una de ellas, se presenta un conjunto de parámetros que describen el tráfico existente en la red y la QoS que se requiere en ella para cada categoría de servicio. En [7] se amplían algunos temas descritos por ITU-T en sus recomendaciones I.371, I.150 e I.356. Algunas de las categorías de servicio del ATM Forum son equivalentes a las capacidades de transferencia de ATM descritas por ITU-T en la Rec. I.371 [4], aunque se usan diferentes nombres: CBR es llamada DBR por ITU-T y VBR se denomina como SBR. En general, la relación entre las categorías de servicio de ATM Forum y las capacidades de transferencia de ITU son:

- ATM Forum distingue entre rt-VBR y nrt-VBR, mientras la Rec. I.371 especifica non-real-time SBR y deja la real-time SBR para estudios futuros.
- La categoría de servicio VBR del ATM Forum no tiene equivalencia en las capacidades de transferencia de I.371.
- En I.371 se especifica parcialmente una capacidad de transferencia ATM llamada ABT que no tiene equivalencia en ATM Forum.
- ABR está completamente especificada por ATM Forum pero sólo lo está parcialmente para ITU-T que la mantiene bajo estudio.
- CT es sólo propuesta por la ITU-T como ATC de flujo controlado a través de créditos de retorno. El ATM-Forum mantiene la decisión de considerar sólo ABR como esquema de velocidad basado en control de flujo.
- Tanto ITU-T como ATM-Forum estandarizan GFR como una forma de dar servicio a aplicaciones que necesitan transferir tramas.

A continuación se definen brevemente las cinco categorías de servicio enunciadas por el ATM Forum*:*

- CBR (Constant Bit Rate): Se emplea para las conexiones que solicitan un tamaño de ancho de banda estático que deberá estar completamente disponible para la aplicación durante todo el tiempo que dure la conexión. El tamaño del ancho de banda se caracteriza por el valor PCR (Peak Cell Ratio). Esta categoría de servicio puede emplearse, tanto para VPCs como para VCCs. El servicio CBR está pensado para soportar aplicaciones en tiempo real con pequeñas variaciones de retardo (voz, vídeo, emulación de circuitos) pero no queda únicamente restringida a estas aplicaciones.
- rt-VBR (real time-Variable Bit Rate) pensada para aplicaciones de tiempo real que requieren mínimo retardo y mínimas variaciones de retardo, apropiadas para aplicaciones de voz y vídeo. Sus fuentes de información pueden entenderse como ráfagas y se espera que transmitan ratios distintos a lo largo del tiempo. Las conexiones de este tipo se caracterizan por los valores PCR, SCR (Sustainable Cell Rate) y MBS (Maximum Burst Size).
- nrt-VBR (non-real-time-Variable Bit Rate) pensada para aplicaciones sin requerimientos de tiempo real y con tráfico a ráfagas. La aplicación tiene un ratio bajo de pérdidas de células y no existen límites de retardo.
- UBR (Unspecified Bit Rate) pensada para aplicaciones sin necesidades de tiempo real ni de bajo retardo ni variaciones de retardo. Son aplicaciones típicas de esta categoría de servicio la transferencia de ficheros y el correo electrónico. Los servicio UBR no permiten especificar ninguna garantía de servicio a sus tráficos de información.

- ABR (Available Bit Rate). En esta categoría de servicio las características de la capa de transferencia ATM, que han sido ofrecidas por la red, pueden cambiar después de establecida la conexión. Se especifican mecanismos de control de flujo que soportan varios tipos de realimentación para controlar el ratio de la fuente en respuesta a los cambios de las características de transferencia de la capa ATM. La realimentación es convenida en la fuente a través de las células específicas de control llamadas RM. Se espera que un sistema final sea capaz de adaptar su tráfico según la realimentación, alcanzando así un bajo ratio de pérdidas y conseguir compartir justamente el ancho de banda de acuerdo a la política de asignación especificada. ABR no está pensada para soportar aplicaciones en tiempo real y por eso no requiere límite de retardo ni variaciones en el retardo. En el establecimiento de la conexión el sistema final especifica a la red el máximo ancho de banda que necesita (PCR) y el mínimo ancho de banda utilizable (MCR). El ancho de banda disponible en la red puede cambiar con el tiempo, pero no será menor al valor de MCR. Destacar que [7] está especialmente dedicada a la clase de servicio ABR, aunque no entra a tratar las problemáticas de mp-mp.

La *Tabla 1.2* muestra algunas de las características principales de las CoS definidas por el ATM-Forum.

TABLA 1.2
CARACTERÍSTICAS DE LAS CLASES DE SERVICIO ATM

Características	CBR	RT-VBR	NRT-VBR	ABR	UBR
Ancho de Banda Garantizado	SI	SI	SI	Opcional	NO
Adecuado para tráfico tiempo real	SI	SI	NO	NO	NO
Adecuado tráfico a ráfagas	NO	NO	SI	SI	SI
Realimentación en congestiones	NO	NO	NO	SI	NO

1.4. PARÁMETROS DE CALIDAD DE SERVICIO

La calidad de servicio, o QoS (Quality of Service), se define en [8] como "*...los resultados globales de funcionamiento de un servicio que determinan el grado de satisfacción del usuario de dicho servicio.*". La Rec. E.800 [8] ha sido posteriormente modificada por la Rec. I.350 [9] limitando los parámetros de calidad de servicio a aquellos que pueden observarse y medirse directamente en el punto en que accede el usuario. Es decir, quedan fuera aquellos parámetros de QoS que son de naturaleza subjetiva o dependen de las opiniones del usuario.

La Rec. I.350 describe también la calidad de funcionamiento de la red, NP (Network Performance) que se mide en términos de parámetros significativos para el proveedor de la red, y que se usan con fines de diseño, configuración, explotación y mantenimiento del sistema. Así, la NP se define independientemente del funcionamiento de los terminales y de la actuación de los usuarios.

Existen, por tanto, diferencias entre los conceptos de QoS y NP. Los parámetros de QoS perceptibles por el usuario configuran el marco para el diseño de redes, pero no son necesariamente utilizables para especificar los requisitos de NP de ciertas conexiones. Del mismo modo, los parámetros de NP determinan al final la QoS obtenida por el usuario, pero no describen necesariamente la calidad de forma significativa para los usuarios. En suma, tanto los parámetros de QoS como los de NP son necesarios, y sus valores deben estar cuantitativamente relacionados para que la red sirva eficazmente a sus usuarios.

Así como en [9] no están claramente descritos los parámetros de QoS, sí presenta varios parámetros primarios genéricos de NP de los que se derivan otra serie de parámetros. Los parámetros primarios son: velocidad de acceso, precisión de acceso, seguridad de acceso, velocidad de transferencia de información, precisión de transferencia de información, seguridad de transferencia de información, velocidad de desvinculación, precisión de desvinculación y seguridad de desvinculación. Cabe destacar que todos estos parámetros, y sus derivados, pueden usarse para establecer parámetros de QoS y NP especificados.

Como podemos ver, desde el punto de vista formal y estándar, existen consideraciones diferentes para el punto de vista del usuario y del proveedor de la red. En nuestro caso, vamos a tratar de forma genérica la QoS sin entrar en diferenciaciones formales, por lo que podemos decir que para nosotros la QoS es la

habilidad para definir o predecir el rendimiento de una red y ofrecer mejores servicios a una CoS específica. La QoS puede ser configurada con un elemento de red que incluya sistema de colas, planificación y caracterización del tráfico. Varias son las técnicas de señalización usadas para coordinar la QoS extremo-extremo entre diferentes redes o elementos de una red. También es destacable el que los criterios de QoS son muy diferentes para cada uno de los tipos de tráfico posibles en ATM, por ejemplo, la QoS para el tráfico multimedia es muy diferente que para el tráfico de datos, voz o vídeo. Incluso, en el caso de tráfico de datos la QoS es diferente para las diversas clases de tráfico de datos. Además, los parámetros de QoS son dependientes del tipo de red: la QoS en una red local no es la misma que en una WAN, ni tampoco son iguales para una red fija o para una red inalámbrica. En este apartado queremos centrarnos en los parámetros generales de QoS de la tecnología ATM, para pasar luego a particularizarlos en cada una de las CoS descritas en el apartado anterior. Posteriormente nos centraremos en los parámetros específicos de QoS de la clase de servicio ABR y UBR que son las más apropiadas para ser soportadas en la arquitectura TAP propuesta en esta tesis.

Tanto en el caso del tráfico de datos, como en el de las aplicaciones multimedia, la noción de QoS es muy importante y está definida como un conjunto de parámetros que representan las propiedades del tráfico. En general, existen los siguientes cuatro parámetros básicos [9] de QoS:

- El rendimiento (throughput*)* es el parámetro más importante y especifica cuántos datos (máximo o en media) son transferidos a través de la red. En general, no es suficiente especificar el ratio en términos de bits por segundo, sino también en unidades de paquetes, ya que el esquema de calidad de servicio debe ser aplicable a varias redes y sistemas de propósito general.
- El parámetro retardo (delay*)* expresa el máximo retardo observado por una unidad de datos en una transmisión extremo-extremo.
- La variabilidad (jitter*)* expresa la variación experimentada entre retardos consecutivos durante la transmisión y procesamiento de datos. El jitter puede amortiguarse con técnicas de *buffering* en los receptores lo que, a su vez, incrementa el retardo extremo-extremo.
- Por otro lado, la fiabilidad (reliability*)* está referida a las pérdidas y corrupciones de datos durante las transferencias.

En realidad, estos cuatro parámetros de QoS aportan a ATM grandes ventajas con respecto a otras tecnologías, sin embargo, existen una extensa serie de parámetros que están directamente relacionados con la QoS. La *Tabla 1.3* muestra algunos de estos parámetros que se usan para caracterizar el tráfico que genera cada una de las fuentes, los cuáles están directamente relacionados con cada una de las CoS ya comentadas.

TABLA 1.3
PARÁMETROS DE TRÁFICO Y DE QOS

Parámetro		Significado
PCR	Peak Cell Rate	Máxima velocidad a la que se envían células
SCR	Sustained Cell Rate	Velocidad media de células a largo plazo
MCR	Minimum Cell Rate	Velocidad de células mínima
CDVT	Cell Delay Variation Tolerance	Máxima fluctuación de retardo de células
CLR	Cell Loss Ratio	Tasa de células perdidas o entregadas con retardo
CTD	Cell Transfer Delay	Tiempo que tarda una célula en llegar extremo-extremo
CDV	Cell Delay Variation	Variación entre los retardos de llegada de células
CER	Cell Error Ratio	Porcentaje de células erróneas que llegan al destino
SECBR	Severely-Errored Cell Block Ratio	Porcentaje de tramas que contienen células erróneas
CMR	Cell Missinsertion Rate	Células entregadas a destino erróneo por errores en cabecera
MBS	Maximum Burst Size	Máximo tamaño de ráfaga permitido
MFS	Maximum Frame Size	Máximo tamaño de trama permitido
IBT	Intrinsic Burst Tolerance	Tolerancia a la aparición de ráfagas
ACR	Allowed Cell Rate	Velocidad máxima de células autorizada a la fuente
ECR	Explicit Cell Rate	Velocidad máxima de células explícitas autorizada a la fuente
BCR	Block Cell Rate	Velocidad pico de célula de bloques

1.5. RELACIONES CALIDAD DE SERVICIO Y CLASES DE SERVICIO ATM

Podemos encontrar la relación o asociación entre QoS y CoS solapando las Recomendaciones I.371 e I.356. De este modo vemos cómo cada una de las CoS están caracterizadas con sus propios parámetros de tráfico que acaban determinando la QoS que cada fuente especifica en el establecimiento de la conexión. En este apartado queremos realizar una exposición del resto de parámetros de QoS no comentados en el punto anterior, relacionándolos con cada una de las CoS de ATM.

Las categorías de servicio ATM suelen definirse en general [7] usando los tres siguientes parámetros de QoS que se negocian entre los extremos de cada comunicación en el proceso de establecimiento de la conexión, que describen características de la red y se miden en los receptores:

- Máximo retardo en las transferencias de células (maxCTD), mide el tiempo medio de propagación de células entre el emisor y el receptor.
- Variación en el retardo de células pico-a-pico (peak-to-peak CDV o jitter), expresa la uniformidad con que el nodo emisor entrega las células al receptor. Los retardos son debidos al tiempo de propagación, al de conmutación y a las congestiones de los conmutadores.
- Ratio de pérdida de células CLR (Cell Loss Ratio), mide la tasa de células perdidas por la red.

Por otro lado, los tres siguientes parámetros de QoS también especifican características de la red y no suelen ser negociables:

- CER (Cell Error Ratio), expresa la tasa de células que llegan al receptor con uno o más bits erróneos.
- SECBR (Severely Errored Cell Block Ratio), es un parámetro relacionado con la transferencia de paquetes que contienen un número *n* de células entre las cuales existe un número de *x* células con errores.
- CMR (Cell missinsertion Rate), mide el número de células por segundo que se entregan a destinatarios erróneos debido a errores producidos y no detectados en las cabeceras de las células.

Respecto a la anterior clasificación se han definido las siguientes clases de *QoS* para cada clase de servicio:

- QoS esfecífica clase 1: Soporta QoS con requerimientos de Clase de servicio A. Está pensada para las actuales líneas digitales privadas.
- QoS esfecífica clase 2: Para QoS de Clase de servicio B. Esta QoS está pensada para empaquetar vídeo y audio en aplicaciones de teleconferencia y multimedia.
- QoS esfecífica clase 3: Para Clases de servicio C. QoS pensada para interconectar protocolos orientados a la conexión como Frame Relay.
- QoS esfecífica clase 4: Para Clase de servicio D y pensada para protocolos no orientados a la conexión como IP y SMDS.

La *Tabla 1.4* muestra las relaciones entre las CoS de ITU-T y los diferentes parámetros de QoS. Cada *X* indica que un parámetro de tráfico es soportado por una CoS determinada. La tabla muestra también las CoS que emplean células RM (Resource Management) como mecanismo de realimentación que genera células informativas en los dos sentidos de la comunicación para conocer el estado de la red.

Llegados a este punto queremos adelantar que en nuestro caso no nos vamos a centrar en ninguna de las CoS estándares ya que empleamos fuentes ON/OFF para realizar nuestras simulaciones. Sin embargo, como ya veremos, este tipo de fuentes también son caracterizadas con parámetros de tráfico muy similares a las especificadas para estas clases de servicio. Más adelante destacaremos también cómo las CoS estándares que mejor se ajustan a nuestra arquitectura son ABR y sobre todo UBR por estar pensada para el transporte de datos. No obstante, la novedosa clase GFR por su orientación al uso de tramas puede ser ajustada también a nuestras propuestas basadas en PDUs. La clase CT, propuesta para LAN también tiene interesantes características para nuestros objetivos de constitución de una red privada virtual.

TABLA 1.4
MAPEO ENTRE CoS Y QoS

Parámetros	DBR	SBR	ABT	ABR	GFR	CT
RMs			X	X		X
PCR	X	X	X	X	X	X
CDV	X	X	X	X	X	X
CLR	X		X	X		X
SCR		X	X			
MBS		X			X	
BCR			X			
MCR				X	X	
ECR				X		
ACR				X		
MFS					X	
IBT					X	

1.6. CONTROL DE CONGESTIÓN

Hemos comentado ya que las células aportan un mecanismo de control de errores basado en el campo HEC. Sin embargo, los errores producidos en las transferencias a través de la red no son el principal problema que pueden experimentar las células. Un problema más grave y menos predecible es el de la congestión, provocado en los conmutadores cuando se les exige un rendimiento superior al que son capaces de ofrecer. Es decir, los conmutadores pueden experimentar congestiones cuando las fuentes de tráfico no cumplen sus contratos de tráfico y producen, individualmente o de forma conjunta, más células por segundo que las que alguno de los conmutadores intermedios es capaz de procesar.

Pues bien, la tecnología ATM tiene solución para los dos tipos de congestión que pueden darse. Por un lado nos encontramos con la congestión a largo plazo provocada por la generación de un tráfico superior al que puede manejar la red y, por otro lado, tenemos la congestión producida a corto plazo que es causada por fuentes que generan tráfico a ráfagas que no mantienen un comportamiento estable en sus parámetros lo que impide su caracterización. Para solventar estos problemas se han propuesto múltiples soluciones que giran en torno a las siguientes ideas:

- Control de admisión. Las características de escalabilidad e integración de tráfico, ya comentadas, aportan importantes ventajas a los usuarios de ATM. Sin embargo, también complican de forma destacable las labores de gestión de la red. Si estamos ante una red capaz de soportar poca velocidad de transferencia puede ser interesante esperar a la aparición de las congestiones para tomar medidas e informar a las fuentes que están produciendo la congestión mediante células RM. Pero como hemos visto antes, no todas las CoS soportan células RM, por lo que no siempre funcionará este mecanismo. Por otro lado, si la congestión se produce en una red de elevado ancho de banda puede que este mecanismo de notificación a la fuente no sea eficiente porque puede ser que se pierdan demasiadas células antes que el aviso llegue a la fuente que está generando el tráfico, lo que puede ser un grave problema en el caso de transmisiones de datos. Sin embargo, cuando lo que se está transfiriendo es tráfico en tiempo real, no interesará efectuar ningún control de congestión mientras la tasa de células perdidas no supere una tasa umbral. Por otro lado, nos encontramos también con que CoS como VBR, CBR y UBR no soportan ningún mecanismo de control de congestión. En vista de estas situaciones quizás sea interesante recordar el comentario de Tanenbaum en [11] "*...un poco de prevención es preferible a una dosis de medicina...*". Es decir, es preferible poner más esfuerzo en evitar que las congestiones aparezcan que en reaccionar cuando ya han aparecido. En esta línea se propone el control de admisión para evitar que se produzcan congestiones. Este mecanismo consiste en que cada fuente debe caracterizar el tráfico que va a generar y especificar sus requerimientos antes de ser admitido por la red. Si la red no puede encontrar un circuito virtual que garantice las necesidades del usuario no permitirá su admisión para evitar que su entrada pueda perturbar el buen estado de la red y de las conexiones que ya han sido admitidas. Esta es la labor realizada por la función CAC que comentamos al inicio de este capítulo.

- Reserva de recursos: Un buen complemento al control de admisión es la posibilidad de reservar los recursos antes de realizar la comunicación y que generalmente se realiza en el establecimiento de la conexión. En realidad, esto hace referencia al contrato de tráfico que se puede firmar con la función CAC, que se encarga de buscar un camino virtual que garantice el contrato de tráfico y, si lo encuentra, lo que hace es reservar los recursos contratados para que no sean tomados por otra conexión. La reserva de recursos puede hacerse respecto a cualquiera de los parámetros de QoS que hemos descrito anteriormente.
- Control basado en velocidad explícita: Esta es la técnica usada por la CoS ABR que a través de células RM generadas con una frecuencia fija permite que los emisores conozcan la velocidad que los conmutadores son capaces de soportar en todo momento. La frecuencia fija de generación de las células RM permite saber cuándo se ha perdido alguna de ellas, de forma que el emisor puede rebajar su tasa cuando ocurre esto ante la sospecha de posibles congestiones. Las células RM que no se pierden llegarán al emisor con el campo ER (Explicit Rate) indicando la velocidad que es capaz de soportar el conmutador más lento de todo el circuito virtual que se está usando. Existen diversas implementaciones de esta técnica y en nuestro caso nos basaremos en una variante de ésta para ofrecer garantía de servicio a las transferencias privilegiadas. Destacamos que nuestro protocolo no se propone para evitar o aminorar las congestiones, ya que para esto ya suponemos que existe algún mecanismo basado en velocidad explícita. La aportación de nuestra tesis es la recuperación de aquellas células que se pierden cuando aparecen las congestiones y que, como hemos dicho antes, pueden ser muchas si estamos ante una red de ancho de banda limitado. Proponemos además la recuperación de las células mediante retransmisiones entre los conmutadores, en lugar de realizarlas extremo-extremo ya que éstas pueden acabar degenerando aún más el mal estado de la red.

1.7. CONCLUSIONES

En este primer capítulo hemos realizado una descripción general de los fundamentos en los que se basa la tecnología ATM poniendo mayor atención en aquellos aspectos que consideramos más importantes para justificar nuestra tesis. De este modo hemos destacado aquellos campos de las cabeceras de las células ATM de mayor interés para nosotros. Igualmente hemos destacado el modelo de referencia arquitectónico que vamos a respetar, pero al que aportaremos nuevas funcionalidades con una modificación de la capa AAL-5. A continuación hemos destacado las clases de servicio destacando que UBR y ABR son las que centran nuestra atención, aunque hemos identificado otras posibilidades en las nuevas clases de servicio. Nuestra intención es soportar QoS garantizada a conexiones privilegiadas por lo que hemos identificado también los parámetros de tráfico de las CoS. Para aportar la garantía de servicio que soporta nuestra arquitectura TAP proponemos solventar los problemas provocados por las situaciones de congestión en los conmutadores, por lo que hemos revisado las posibilidades que tiene la tecnología para evitar las congestiones. No obstante, hemos podido comprobar cómo, aunque existen ya mecanismos de control de congestión, las congestiones pueden aparecer de forma indeterminista y en esos momentos es cuando tendrá sentido nuestro protocolo de recuperación de células (PDUs) punto-a-punto en lugar de realizarlas extremo-a-extremo. Todas estas consideraciones, y varias más, serán descritas oportunamente en los capítulos siguientes.

REFERENCIAS

[1] ____, Draft Recommendation I.361, "B-ISDN ATM Layer Specification", *CCITT Study Group XVIII*, Geneva, May 1990.

[2] ____, Draft Recommendation I.363, "B-ISDN ATM Layer Specification", *CCITT Study Group XVIII*, Geneva, May 1990.

[3] ____ Rec. I.363.5, "Capa de adaptación del modo de transferencia asíncrono tipo 5," *ITU-T*, (08/1996).

[4] ____ Rec. I.371, "Control de tráfico y control de congestión en la RDSI-BA", *ITU-TS*, (Aug. 1996).

[5] S. Van Luinen, Z. Budrikis, and A. Cantoni, "The Controlled Cell Transfer Capability", *ACM SIGCOMM Computer Communication Review*, pp. 55-71, (1997).

[6] ____ **"**General Network aspects", *ITU-T SG/13 plenary meeting, ITU-T*, (June 1998).

[7] ____ "Traffic Management Specification Version 4.0," *ATM Forum Technical Committee, ATM Forum Document af-tm-0056.000*, (April 1996).

[8] ____ Rec. E.800 "Terms and Definitions Related to the Quality of Telecommunicactions Services", *CCITT*, (1988).

[9] ____ Rec. I.350 “Aspectos generales de calidad de servicio y de calidad de funcionamiento en las redes digitales incluidas las redes digitales de servicios integrados”, *ITU-T*, (1993).

[10] Steinmetz, R.,and Wolf, L.C.,“Quality of Service: Where are We?,” *IWQOS’97*, pp.211-221, (1997).

[11] Tanenbaum, A. S.“*Computer Networks”*, 3ª Ed.Upper Saddle River. NJ. *Prentice Hall, Inc*., 1996.

CAPÍTULO 2

TAXONOMÍA DE ARQUITECTURAS Y PROTOCOLOS PARA REDES ATM

2.1. INTRODUCCIÓN

La tecnología ATM continua evolucionando y siendo fuente de interesantes y novedosas propuestas. El desarrollo de avanzados protocolos de comunicaciones es uno de los campos de investigación más activo, con la aspiración de ofrecer el adecuado soporte a las nuevas aplicaciones adaptadas a las clases de servicio nativas ATM. Los protocolos son los responsables, entre otras funciones, de garantizar la QoS demandada por los usuarios de avanzados servicios multimedia. En este contexto son aspectos clave los relativos a los protocolos nativos ATM, así como las características multicast, la escalabilidad y la fiabilidad.

La actual demanda de aplicaciones relacionadas con información multimedia, como son la video-conferencia, audio-conferencia, video bajo demanda (VoD) o sistemas colaborativos (pizarras compartidas, teletrabajo, telemedicina, etc.) y su coexistencia con aplicaciones más clásicas (bases de datos, transferencias de ficheros, WWW, etc.), requiere de tecnologías de comunicaciones capaces de ofrecer elevadas prestaciones. Estas prestaciones están directamente relacionadas con la QoS y, más concretamente, con conceptos parametrizables como el ancho de banda y la velocidad de transmisión (throughput), el retardo de las transferencias (delay); la variabilidad en los retardos (jitter); la fiabilidad (reliability) de las transmisiones; las características de multidifusión a grupos dispersos de usuarios (multicast) y la posibilidad de gestionar múltiples clases de servicio o flujos de información en redes multiclass.

Para que las nuevas tecnologías en comunicaciones puedan ofrecer estas características es necesario revisar, potenciar y ampliar las actuales arquitecturas, servicios y protocolos de comunicaciones. En los últimos cinco años, las investigaciones en el campo de ATM están dado lugar a importantes propuestas cuyo principal objetivo es ofrecer a las aplicaciones más demandadas algunas, todas o superiores características a las citadas anteriormente.

Con este objetivo, nuestra tesis propone una nueva arquitectura que soporta un protocolo que aporta transferencias garantizadas a una serie de conexiones privilegiadas. Consideramos oportuno, por tanto, presentar en este capítulo los conceptos y propuestas más novedosas en materia de ingeniería de protocolos en el contexto de ATM para centrar este dinámico, complejo y sofisticado campo en el que nos vamos a mover. De este modo, revisaremos algunos de los más importantes conceptos, técnicas, ideas y mecanismos en materia de protocolos de altas prestaciones para redes de tecnología ATM. El principal objetivo de este capítulo es ofrecer una visión general actualizada, no extensiva ni profunda, de los protocolos propuestos y en desarrollo para dotar a las diversas clases de servicio ATM de las citadas características de QoS que aporten las potenciales elevadas prestaciones que la tecnología ATM es capaz de ofrecer.

Este capítulo contiene muchas de las lecciones aprendidas en el proceso de documentación e investigación de esta tesis, y nos permitirá centrar aspectos importantes que han acabado dando lugar a la propuesta de la arquitectura TAP (Trusted and Active Protocol). Así, comenzamos con una revisión de los conceptos más importantes en el campo de los protocolos de comunicaciones para conocer los objetivos de las nuevas propuestas comparativamente con los protocolos estándares. A continuación analizamos el concepto de protocolo nativo centrado en el contexto de las redes ATM. En el apartado siguiente describimos

las propuestas más interesante en materia de protocolos de transporte y protocolos multicast. Se comentan después las transferencias multicast como un importante objetivo para una tecnología orientada a la conexión como es ATM. Las transferencias punto-multipunto y multipunto-multipunto están generando investigaciones que han dado lugar a algunos protocolos multicast situados en la capa de transporte de estas redes de alta velocidad. La sección siguiente resume las diferentes propuestas existentes para el soporte del tráfico IP sobre la tecnología ATM, en un intento por integrar los flujos de información más extendidos sobre la tecnología más avanzada. Concluiremos destacando los campos de investigación abiertos recientemente como son las redes activas y los agentes software que en capítulos siguientes serán comentados con mayor profundidad ya que son el fundamento de muchas de las aportaciones de esta tesis. El dinámico campo del desarrollo de los protocolos nos lleva a no ser exhaustivos en nuestro estudio, por lo que no citamos todos los protocolos, ni presentamos todos los conceptos ni analizamos todas las soluciones existentes en la literatura.

2.2. CONCEPTOS CLAVE

La tecnología ATM ofrece importantes características como la integración de servicios, el elevado rendimiento y la escalabilidad. Las aplicaciones multimedia son el objetivo de las redes ATM ya que éstas requieren de características como la garantía de QoS y el soporte multicast. Las redes ATM se caracterizan también por congestiones indeterministas en los conmutadores causadas por ráfagas esporádicas provenientes de varias fuentes, y por frecuentes, inesperados y repentinos cambios en el ancho de banda disponible debidos a llegadas y salidas de conexiones ABR, CBR, VBR o UBR. ATM es, por tanto, una tecnología de comunicaciones mucho más sofisticada y compleja que otras como X.25, IP, Frame Relay, etc. Las investigaciones constantes en este campo se encargan de ofrecer nuevas características y conceptos, algunos de los cuales van a ser revisados brevemente en este apartado ya que serán empleados en las siguientes secciones y capítulos.

2.2.1. MODO ATM NATIVO

Las aplicaciones nativas ATM están específicamente pensadas para usar la tecnología ATM y para explotar al máximo sus especiales características. Los protocolos nativos se encargan, por tanto, de ofrecer esas características intrínsecas de las redes de tecnología ATM (soporte de QoS, señalización, direccionamiento, etc.) a las aplicaciones nativas ATM (VoD, pizarras compartidas, video-conferencia...). No obstante, existen también activas investigaciones para conseguir soportar sobre redes ATM aplicaciones no nativas ATM desarrolladas para otras tecnologías (IP, Frame Relay, SMDS...).

En [1], el termino *native ATM services* define servicios ATM específicos disponibles para el software y hardware residentes en dispositivos de usuario UNI ATM. Por tanto, el programador de aplicaciones dispone de nuevos servicios entre los que se pueden destacar los siguientes:

- Transferencias de datos (fiables o no) usando la capa ATM y varias capas de adaptación (AAL).
- Disponibilidad para usar y ofrecer circuitos virtuales conmutados (SVC) y circuitos virtuales permanentes (PVC).
- Consideraciones relativas a la gestión de tráfico (clases de servicio, garantías de QoS, etc.).
- Posibilidad de distribución de conexiones.
- Posibilidad para participar localmente en la administración de la red (protocolos ILMI y OAM).

El propósito de los servicios nativos ATM es ofrecer el acceso a las CoS o a las características de QoS. Estos servicios nativos también ofrecen soporte a un amplio y heterogéneo rango de flujos con diversas propiedades y requerimientos recomendados en [1].

Los protocolos de transferencia nativos ATM gestionan la señalización UNI para establecer los SVC, configurar PVC y mapear los perfiles de QoS en la correspondiente CoS. Los protocolos nativos también realizan funciones clásicas como las de transporte, mecanismos de control de errores, transferencia de datos, y controles de flujo y de congestión.

En la referencia [1] se especifica la definición semántica de los servicios y consideramos que es útil contrastar esta semántica con las redes ATM actuales que usan TCP como capa de transporte, e IP-over-ATM como capa de red. Planteamiento éste que puede considerarse poco adecuado [2] por estas razones:

- Las redes IP no garantizan la QoS extremo-extremo ofrecida por las redes ATM a circuitos individuales. IP multiplexa múltiples conexiones de transporte con distintos requerimientos de QoS en VC simples.

- TCP no soporta células RM (Resource Management) ABR y, como consecuencia, no puede usar directamente las garantías de QoS ofrecidas por la red.
- ATM Adaptation Layer 5 (AAL-5) realiza labores de checksum para detectar corrupción de datos. TCP también realiza estas labores (redundantes con AAL-5) costosas en overhead (cada byte de un paquete debe ser chequeado).
- TCP e IP son la representación de un grupo de protocolos anteriores a ATM y que ya han experimentado determinados arreglos y evoluciones, lo mismo que las aplicaciones que los emplean, lo que acaba dando, en algunos casos, a inadecuados comportamientos por la evolución independiente de ambas tecnologías.

Estos y otros problemas son eliminados usando pilas de protocolos en modo nativo ATM que son revisados en el siguiente apartado. La *Figura 2.1* muestra el modelo de referencia para servicios nativos ATM [1].

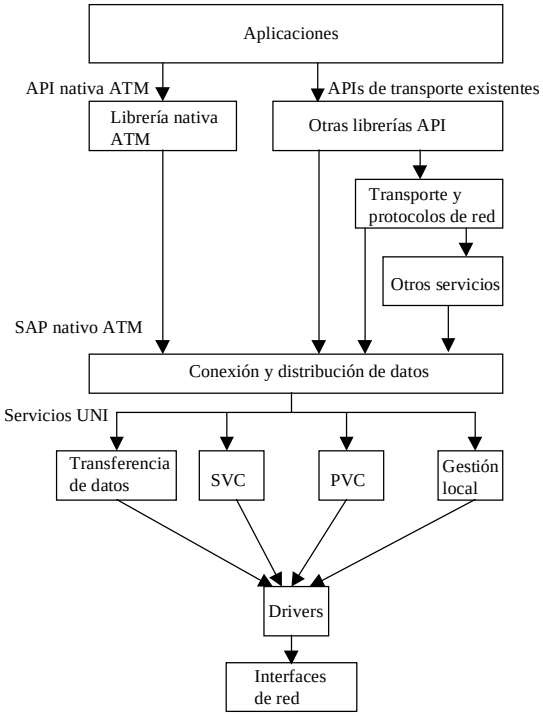

Figura 2.1. *Modelo de referencia para servicios nativos ATM* [1]

Para concluir pueden argumentarse las siguientes razones [3] para justificar el desarrollo de pilas de protocolos nativos ATM en las que se basa también nuestra propuesta TAP:

- En la actualidad [3] existen aplicaciones pensadas para explotar avanzados servicios usando tecnología ATM y también existen otras aplicaciones más antiguas y no nativas. Este escenario implica cambiar las aplicaciones o proponer nuevas pilas de protocolos nativos ATM.
- La encapsulación consecutiva de paquetes genera problemas de overhead y funciones redundantes como se ha argumentado anteriormente.
- La limitación de recursos en los sistemas finales es otra importante motivación para usar pilas de protocolos nativos y ligeros.
- La QoS ofrecida por el modo nativo es aprovechada por los usuarios para demandar recursos a los proveedores de servicios en redes privadas. Los proveedores de servicios públicos disfrutan también de estas ventajas.
- ATM, RDSI y la telefonía ofrecen un esquema de direccionamiento universal basado en NSAP/E.164 el cual es capaz de enrutar tráfico de forma nativa. Por tanto, aunque ATM dispone de protocolos nativos con direccionamiento intrínseco, estructurado y jerárquico, éste no es aprovechado por las

aplicaciones que están basadas en IP. El esquema de direccionamiento ATM es una de las principales dificultades en los protocolos propuestos como nativos [2,3]

2.2.2. ESCALABILIDAD

ATM Forum ha normalizado protocolos de routing y señalización para PNNI (Private Network-Network Interface) para conseguir la escalabilidad de las redes ATM. Los actuales estándares proponen esquemas de señalización y routing *punto-a-punto*. La referencia [4] identifica el problema asociado con los protocolos multipunto-a-multipunto y propone soluciones a este problema.

Los componentes diseñados para ofrecer QoS deben ser escalables, es decir, deben poder ser usados aunque sean aplicados a muy larga escala. Con respecto a las aplicaciones multimedia, la escalabilidad tiene, como poco, dos aspectos clave [5]:

- Escalabilidad con respecto al número de participantes en una aplicación: debe ser posible transmitir un flujo multicast a un, potencialmente, muy largo número de participantes. Los protocolos Internet IP-multicat y ST-2+ [5] soportan este requerimiento. RSVP [6] también soporta un cierto nivel de escalabilidad, que puede verse comprometida o ser discutible y matizable en el caso de que existan varios miles de participantes.
- Escalabilidad con respecto al número de aplicaciones concurrentes: debería ser posible soportar muchas aplicaciones independientes y, por consiguiente, cientos de flujos simultáneos.

En la actualidad, parece que es más importante la escalabilidad referida a una sola aplicación, pero en el futuro todo dependerá de los requerimientos de cada situación particular.

2.2.3. TRANSFERENCIAS MULTICAST

Los dos modos clásicos de transferencia de paquetes entre nodos son los conocidos unicast y multicast [7,8]. En las transferencias unicast un nodo envía paquetes a un único nodo receptor. Las comunicaciones multicast las constituyen un nodo fuente que realiza una sola operación atómica (sólo se envía una copia de cada paquete sobre un enlace) a un grupo multicast de más de un participante (receptores) no necesariamente en la misma red. Una forma eficiente de hacer esto es construir un árbol de distribución multicast optimizado en coste. El multicasting se usa en aplicaciones multimedia para proveer de datos, audio y video a grupos dispersos de usuarios. El conjunto de participantes puede ser fijo o cambiar dinámicamente durante el curso de la sesión. El broadcast es un caso especial de multicast en el cual los paquetes son transferidos a todos los nodos conectados a una red.

ATM es considerada aún como una tecnología emergente diseñada para ser usada por aplicaciones de datos, audio y video, lo que requiere un buen comportamiento de las transferencias unicast y multicast. User Network Interface (UNI 3.0) para ATM define conexiones punto-a-multipunto, y las conexiones multipunto-a-multipunto sólo pueden ser obtenidas de las dos siguientes formas:

- El primer esquema consiste en configurar *N* conexiones punto-a-multipunto para conseguir conectar todos los nodos en una topología completamente mallada todos-con-todos. Aunque esta topología ofrece conexiones multipunto-a-multipunto, hay que destacar que no escala bien cuando el número de participantes es elevado.
- Una alternativa al anterior esquema es el uso de un servidor que actúa a modo de raíz en el árbol multipunto. Este método sólo requiere un nodo raíz para almacenar información, pero la desventaja de este método son las potenciales congestiones en el servidor cuando debe encargarse de envíos y retransmisiones de las conexiones multipunto-a-multipunto.

Para solventar las limitaciones de UNI 3.0 y UNI 3.1 [18] que soportan conexiones uno-a-muchos, pero no directamente (nativamente) conexiones muchos-a-muchos, y ofrecer a ATM verdadero servicio multicast, ATM Forum, ITU-T e IETF han realizado varias propuestas al actual mecanismo de señalización ATM (UNI 3.1, UNI 4.0), [9-22,70].

2.3. PROTOCOLOS NATIVOS ATM

ATM Forum ha definido las especificaciones [5,23] y también existen importantes investigaciones [1,24] en torno a los protocolos nativos ATM. En esta sección se muestran las propuestas más importantes y actuales en materia de protocolos nativos ATM [2, 3, 24-34] que, a su vez, están sirviendo de base para nuevas investigaciones.

Las referencias [2, 26] presentan el diseño, implementación y comportamiento de una pila en modo nativo ATM y contrasta la semántica de su capa de transporte con TCP. Este trabajo es diferente a IP-over-ATM, y justifica el uso de la pila nativa ATM para solventar automáticamente los siguientes problemas:

- IP-over-ATM no ofrece garantía de QoS pues sus aplicaciones sólo "ven" la interfaz IP.
- El núcleo de los sistemas operativos y los sistemas finales son sobrecargados con considerable complejidad pues el subsistema IP-over-ATM debe encargarse de las peticiones de la señalización.
- IP-over-ATM debe emular el routing IP sobre conexiones punto-a-punto de la red ATM, lo que supone pagar un elevado precio en prestaciones.

La capa de transporte propuesta en [2,26] ha sido construida para minimizar el overhead y para obtener ventaja de AAL-5. Ofrece, entre otras características, la entrega fiable y no fiable de datos con control de flujo. Este protocolo de transporte es ampliado en la sección 4.

La referencia [3] presenta N^3 (Native Non-broadcasting medium access Networking) un protocolo de transporte ligero y nativo ATM. La pila N^3 se ha diseñado para ofrecer servicios multimedia a comunidades residenciales. El documento describe una arquitectura capaz de ofrecer aplicaciones nativas ATM. La pila de protocolo de transporte N^3 se basa en implementaciones previas [2,26] y, además, aporta otras importantes novedades. Los principales componentes de N^3 incluyen una API sockets ATM nativa, un protocolo de transporte ATM y un servicio de nombres ATM (ver *Figura 2.2*).

El protocolo de transporte ATM propuesto en [3] ofrece soporte para tres diferentes tipos de servicio: entrega garantizada (mecanismos de retransmisión), velocidad garantizada (mecanismo leaky bucket) y servicio best-effort. Otro objetivo principal de la pila ATM es su compatibilidad con aplicaciones tradicionales sin necesidad de recompilaciones.

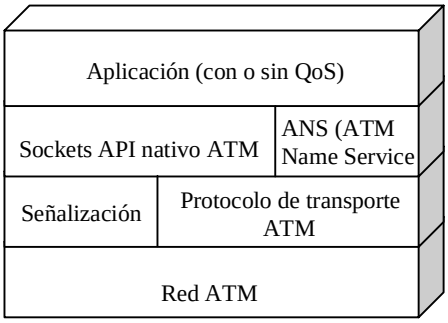

Figura 2.2. *Visión de alto nivel de la pila de protocolos N^3* [3]

El trabajo [27] presenta una arquitectura de servicio ATM en modo nativo ATM capaz de ofrecer a las aplicaciones nativas ATM acceso completo a las CoS ATM. Los elementos de la arquitectura propuesta se responsabilizan de las transferencias eficientes de datos sobre ATM, del control de errores extremo-extremo, del control de flujo y congestión de la transferencia de datagramas y de la multiplexación de VC. La referencia [27] introduce los componentes de la arquitectura, sus funcionalidades y capacidades. Los mayores esfuerzos del protocolo se centran en maximizar la efectividad del rendimiento *extremo-extremo* en canales de datos que usan la CoS UBR.

La *Figura 2.3* presenta los elementos de Native Mode Service Architecture donde el Flow Management es el componente más importante. El Flow Management se responsabiliza de manipular los flujos de datos desde y hasta la red vía la interfaz AAL. La segmentación, el reensamblado y el control de errores es también realizada por esta entidad. Para las CoS CBR, VBR y ABR se emplea un sencillo esquema de control llamado Back-Pressure Flow. Para servicios UBR se emplea un control de congestión y de flujo extremo-extremo más complejo.

Por otro lado, [25] describe la semántica de una pila de protocolos y explora una nueva arquitectura de protocolo adaptada a la tecnología ATM y a las aplicaciones multimedia. El diseño está basado en tres principios básicos: separación de flujos de control y de datos; minimización del overhead y de la duplicidad de funciones; y acceso de las aplicaciones al nivel ATM con garantías de QoS.

La idea es mezclar el soporte nativo ATM en la estructura existente del protocolo (*Figura 2.4*) que muestra dos caminos separados en el protocolo: la familia nativa ATM y la familia del protocolo IP. Las

aplicaciones que tienen acceso transparente a la red ATM usan la familia del protocolo PF_INET. El mapeo de IP en ATM es gestionado por la interfaz de red ATM (IF_ATM) usando el protocolo IP-over-ATM.

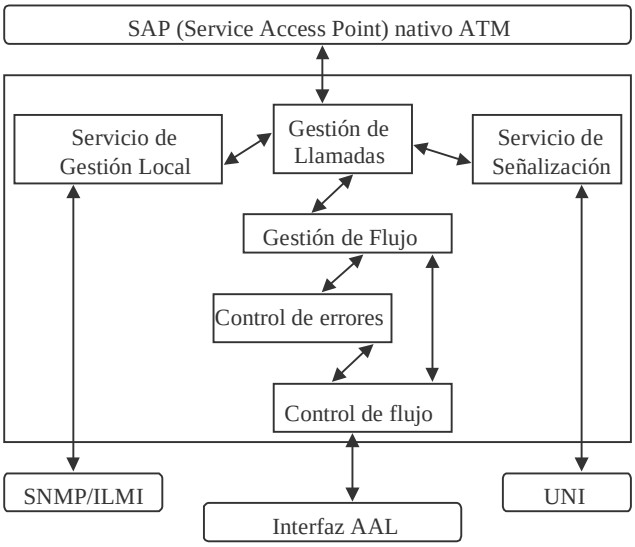

Figura 2.3. *Arquitectura de servicio nativo ATM* [27]

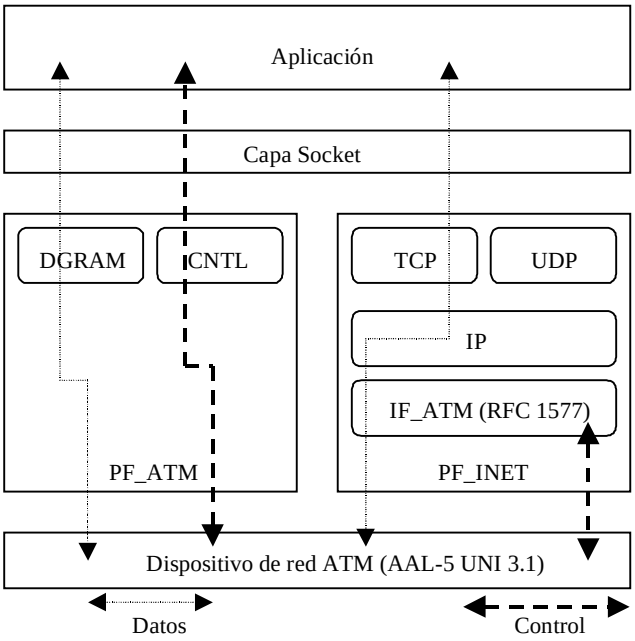

Figura 2.4. *Descripción de alto nivel de la arquitectura de protocolo* [25]

La interfaz Native ATM está constituida por la familia de protocolos PF_ATM que es directamente soportada encima del dispositivo de red ATM sobrepasando la capa interfaz de red. El módulo CNTL abre una conexión de señalización con el dispositivo ATM y establece una gestión de las llamadas de mensajes de configuración.

PF_ATM separa flujos de datos y de control para aliviar el límite de rendimiento en las comunicaciones. Esto permite a los mecanismos de control de tráfico ser rápidos y sencillos, mientras los mecanismos de control pueden ser tan complicados como sea necesario. Esta separación permite también que los dispositivos puedan estar en los puntos finales de una conexión.

La interfaz PF_ATM aporta a las aplicaciones acceso directo a la capa de enlace ATM y extiende las

garantías de QoS a los puntos extremos de la comunicación.

Un segundo prototipo [25] es diseñado e implementado con dos objetivos principales para la mejora de la pila nativa ATM:

- Optimizar los caminos de datos entre los adaptadores ATM aprovechando la separación de flujos de control y de datos y la capacidad de gestión de datos específicos de las conexiones de la pila ATM.
- Optimizar el procesamiento de overheads usando una pila de protocolos nativo ATM en lugar de UDP/IP.

La referencia [33] introduce CONGRESS (CONnection oriented Group-address RESolution Service), otro eficiente protocolo nativo ATM para la resolución y gestión de direcciones de grupos multicast en una red ATM. El servicio CONGRESS resuelve direcciones de grupo multicast y mantiene los miembros pertenecientes a esos grupos para uso de las aplicaciones. CONGRESS sirve de soporte al servicio IMSS [32] presentado en la sección 7, y ofrece escalabilidad con su diseño basado en los dos siguientes principios:

- Diseño jerárquico: los servicios del protocolo son ofrecidos a las aplicaciones por múltiples servidores organizados jerárquicamente.
- No inundación: se evita la inundación de la WAN en cada cambio de grupos multicast.

La propuesta descrita en [34] presenta kStack, una nueva capa de transporte nativa ATM en el espacio de usuario con soporte de QoS. Esta implementación sobre Unix y Windows NT está basada y es compatible con los trabajos originales de Ahuja, Keshav y Saran [2, 26] Native-Mode ATM Stack comentados anteriormente. El protocolo kStack es similar al original, pero se ha modificado sustancialmente en los siguientes aspectos:

- Ha sido implementado en el espacio de usuario.
- Se ha ampliado, implementando una capa de transporte con QoS.
- Se ha añadido un módulo que monitoriza la QoS extremo-a-extremo.

Kstack es por tanto una implementación en el espacio de usuario de una pila de protocolos en modo nativo ATM sobre la capa de transporte. Soporta QoS para monitorización y adaptación del nivel de aplicación. La QoS es monitorizada para cada conexión extremo-a-extremo e independientemente del resto de conexiones.

2.4. PROTOCOLOS DE TRANSPORTE PARA REDES ATM

El crecimiento de las redes ATM viene motivado, en parte, por la demanda de servicios multimedia para grupos dispersos de usuarios. El tráfico multicast tiene características particulares descritas para ATM en UNI 4.0 [22] y anteriores [2,3,18]. La distribución de información punto-a-multipunto (uno-a-muchos), multipunto-a-punto (muchos-a-uno) o multipunto-a-multipunto (muchos-a-muchos) es un objetivo básico propuesto por varios protocolos y arquitecturas ATM. Estos protocolos y arquitecturas ofrecen el soporte multimedia y/o multicast como audio-conferencia, video-conferencia, trabajos colaborativos o VoD.

En los últimos cinco años ha habido numerosos e interesantes intentos por diseñar protocolos de transporte. La referencia [31] presenta un completo, y ya clásico, resumen de las propuestas más interesantes en materia de protocolos de transporte para redes de alta velocidad (mayoritariamente IP en el artículo). Además, la referencia [35] ofrece una interesante visión de las características más importantes en los protocolos de elevada velocidad. Son estudiadas también diversas arquitecturas de protocolos y varias técnicas de implementación.

Los protocolos de transporte de elevada velocidad son fuente de activas investigaciones y están en constante evolución desde hace más de dos décadas, lo que nos impide ser exhaustivo en este resumen.

Uno de los componentes del ámbito de las comunicaciones que ha recibido mayor atención es la capa de transporte, la cuarta capa del *OSI-RM* de los protocolos de comunicaciones. TCP e ISO TP4 son los dos más populares protocolos de transporte.

Centrándonos más concretamente en el ámbito de ATM, la literatura presenta varios protocolos de transporte y arquitecturas para redes de alta velocidad. A continuación se revisan resumidamente los mas representativos.

La propuesta [25] ofrece una excelente y didáctica arquitectura de pila de protocolos para aplicaciones

multimedia en modo nativo ATM. Esta arquitectura de protocolo ha sido ya descrita brevemente en la sección anterior.

Los trabajos [2 y 26] presentan el diseño, implementación y comportamiento de un protocolo situado en la capa de transporte ATM. Esta es una brillante propuesta en la que se puede destacar que la pila ATM está formada por tres entidades principales: las entidades de aplicación y señalización en el espacio de usuario y la entidad de transporte dentro del kernel (*Figura 2.5*).

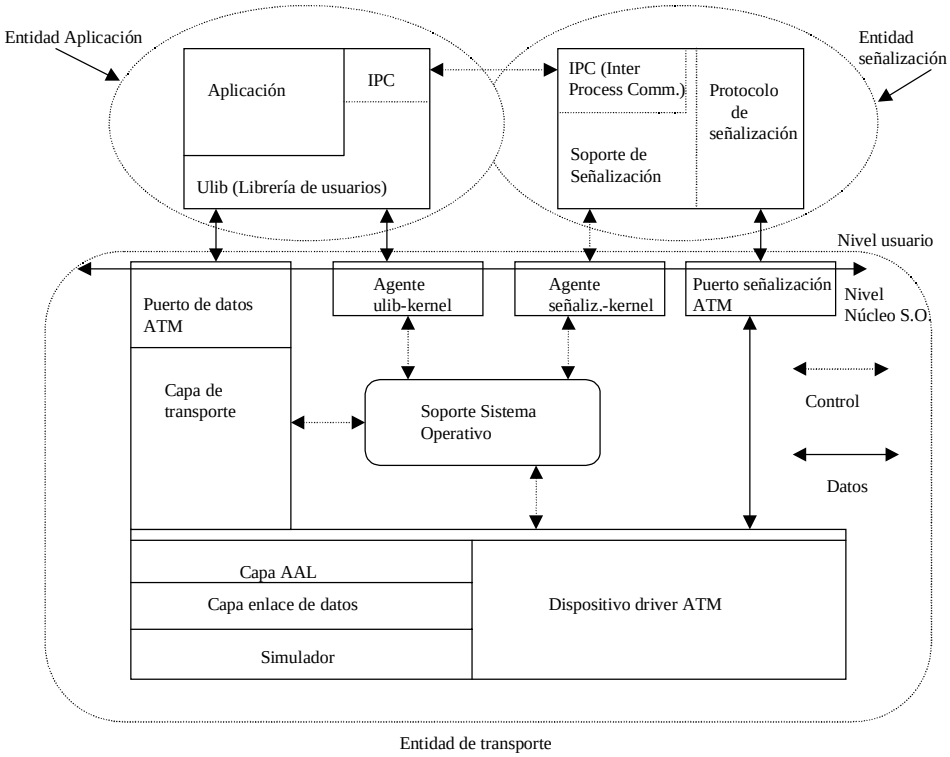

Figura 2.5. *Componentes de la pila ATM (descripción abstracta y entidades)* [2, 26]

La entidad transporte, dentro del núcleo del sistema operativo, se responsabiliza de la transferencia de datos a través del kernel del sistema operativo bajando hasta los adaptadores que ofrecen transporte AAL-5. También realiza llamadas de admisión y asigna los recursos a los VCI para garantizar QoS (ancho de banda y retardo).

La entidad transporte consiste en tres componentes: la capa de transporte, el driver y un módulo de soporte para el sistema operativo. La capa de transporte ofrece circuitos virtuales simplex, control de errores y control de flujo. En suma, segmenta los buffers de la capa de aplicación en TPDUs (Transport Protocol Data Unit) y el reensamblado es realizado por el nodo receptor.

La TPDU es la unidad de transmisión y cada TPDU corresponde a una trama AAL-5 la cual es, como mucho, de 64 Kbytes de larga. Si el mensaje de usuario es mayor de este tamaño, éste debe ser fragmentado por la capa de transporte que será reensamblado después por el receptor. La unidad de detección de error y de retransmisión es también la TPDU para maximizar la eficiencia de las retransmisiones y para minimizar el overhead por segmentación y reensamblado.

Para una conexión fiable, la pérdida o corrupción de datos debe ser retransmitida. Esto es hecho en la capa de transporte usando un novedoso esquema de retransmisión descrito en el artículo. La capa de transporte no realiza checksumming pues es ya desempeñado por la capa AAL.

La *Tabla 2.1* muestra un conjunto de nueve básicos servicios ortogonales que pueden ser combinados para obtener los requerimientos de determinadas aplicaciones. Actualmente, la capa de transporte referenciada en [2,26] soporta tres clases de servicio o combinación de servicios. La marca **X** indica el servicio básico soportado en cada clase de servicio general.

TABLA 2.1. *SERVICIOS ORTOGONALES Y CLASES DE SERVICIO*

	Clases de servicio		
Servicios	**Servicio de comportamiento garantizado**	**Servicio fiable**	**Servicio *Best effort***
- Transferencia Simplex de datos	X	X	X
- Control de errores	-	Detección de errores,*timeouts*, retransmisiones	No existente (ofrecido por AAL-5)
- Bucle abierto - Control de flujo realimentado	No existente -	- X	- No existente
- Tamaño de mensaje ilimitado	X	X	X
- Elección de blocking - Aplicación Non-blocking	X X	X X	X X
- Elección de byte stream - Semántica transferencia de mensajes	X X	X X	X X
- Garantías de QoS	Requerimientos de ancho de banda	No soportado	No soportado
- Reserva de recursos	X	No existente	No existente
- Transferencias Multicast	X	No soportado	Para servicios no fiables

La capa de transporte es esencialmente completa y operativa, pero sus autores destacan que el esquema tiene un problema de justicia debido a las limitaciones del hardware usado. También proponen nuevas áreas para la investigación en la implementación de Resource Manager y en la posibilidad de que el protocolo ofrezca servicio multicast fiable.

En la referencia [36] SHiPP (Super High speed Protocol Processor) se propone una arquitectura programable para el procesamiento de protocolos que incrementa la velocidad mediante técnicas de procesamiento paralelo. Esto es conseguido ejecutando varios procesos hardware en paralelo y cada proceso hardware también ejecuta otro número de pequeñas funciones en paralelo. En este trabajo, SSCOP [37] es usado como un protocolo *benchmark* para SHiPP.

El trabajo [38] resuelve el soporte de HPDC (High Performance Distributed Computing) sobre redes ATM. Los autores sugieren modificaciones en los mecanismos de recuperación de pérdidas del protocolo estándar SSCOP [37] y demuestran que el resultado ofrece baja latencia, eficiente recuperación de células perdidas y es tan robusto como el estándar SSCOP.

Por último, la referencia [39] presenta un fiable y adaptativo protocolo de transporte para soporte de video VBR. Este protocolo minimiza el retardo extremo-extremo y la variabilidad en el retardo de tramas; minimiza los requerimientos de buffers y garantiza que los flujos de video VBR no experimenten pérdidas. Los autores demuestran la adaptabilidad de este protocolo y cómo éste minimiza el retardo y su variabilidad extremo-extremo presentando un algoritmo justo de asignación de buffer/ancho de banda.

2.5. PROTOLOS MULTIPOINT

Comenzamos destacando la referencia [40] como trabajo que revisa y compara los protocolos de transporte *multicast* más importantes en Internet como MTP-2, XTP, RTP, SRM, RAMP, RMTP, MFTP, STORM, etc. IETF e IRTF (como ITU-T y ATM Forum) también impulsan una importante actividad en este campo. La referencia [40] revisa los más representativos protocolos multicast y los clasifica de acuerdo a la taxonomía de varias características (propagación de datos, mecanismos de fiabilidad, retransmisiones, control de congestión y de flujo, gestión de grupos multicast, etc.). En Internet los mecanismos efectivos de control de congestión son una de las prioridades en las investigaciones de las transferencias multicast fiables. Los mecanismos de seguridad y las técnicas escalables de recuperación de errores son algunos de los aspectos actualmente en estudio en el campo de los protocolos de transporte multicast.

Hasta este punto hemos revisado algunos protocolos de transporte ATM y hemos citado sus más importantes características. En esta sección comentaremos los problemas asociados a las transferencias multicast uno-a-muchos o muchos-a-muchos. Actualmente no existen excesivas propuestas en esta importante faceta para ATM, pero vamos a resumir algunas de las más interesantes en los siguientes párrafos.

SMART (shared many-to-many ATM reservations) [14] es un protocolo para controlar un árbol ATM multicast compartido soportando comunicaciones muchos-a-muchos (many-to-many). Esta propuesta tiene

importantes características como que: reside completamente en la capa ATM y no requiere ningún servidor; soporta uno o varios VCC (y también VPC) cuyo número es libremente configurado y es independiente del número de puntos finales; usa el concepto de bloques de datos como en la clase de servicio ABT [20,23,41-49] y también permite VCC de las clases CBR, VBR o UBR; el protocolo garantiza que no existen puntos de interrelación en los VCC del árbol; son respetadas las garantías del contrato de tráfico asociado con los VCC, etc.

SMART puede ser entendido como un protocolo completamente distribuido para coordinar la distribución de los VPI/VCI.

Para solventar las conocidas dificultades debidas al soporte y uso de muchos-a-muchos VCC, SMART usa el mecanismo de Cell Interleaving (sobre un VCC muchos-a-muchos, las células de datos desde diferentes fuentes pueden llegar intercaladas a un destinatario) y también Demand Sharing (los recursos asignados a conexiones muchos-a-muchos son dinámicamente compartidas entre todas las fuentes potenciales.

El artículo [14] describe el protocolo SMART formal e informalmente, y propone completas pruebas de corrección y un detallado análisis de comportamiento para estudios futuros. También son sugeridas otras investigaciones como son: ofrecer justicia en los accesos a los árboles multicast; investigar las células RM periódicamente para aliviar las congestiones en la red o disminuir el tiempo de acceso del usuario a los árboles de distribución multicast; análisis de las células RM dentro de cada VCC o fuera enviando todas las células RM en un VCC dedicado.

En [4] se presenta MWAX un algoritmo dinámico y escalable para routing multicast en el marco PNNI de redes ATM. Los autores han identificado el problema para conseguir la escalabilidad con protocolos multipunto-a-multipunto y se proponen soluciones para este problema. El artículo describe un esquema jerárquico basado en CBT para incorporar routing multipunto-a-multipunto en PNNI. En el algoritmo los nodos core actúan como participantes pasivos para eliminar la dependencia en la selección de estos nodos. Con un mecanismo de backup se consigue un algoritmo tolerante a fallos en los nodos *core*, lo cual puede ser fácilmente extendido para incorporar QoS en el routing multicast. El protocolo-algoritmo MWAS es recursivo, esto es, el mismo protocolo es ejecutado en cada nivel de la jerarquía.

SEAM (Scalable and Efficient ATM Multicast) [50] propone una arquitectura escalable, eficiente y multicast multipunto-a-multipunto para redes ATM que usa un sólo VC para un grupo multicast de múltiples emisores y receptores y todo ello sin realizar cambios en la capa AAL-5 de ATM. Esta propuesta permite a los grupos multicast aprovechar el soporte de QoS y la escalabilidad del ancho de banda. También realiza aportaciones para conseguir soportar IP multicast sobre redes troncales ATM.

SEAM usa un sólo árbol de distribución compartido para todos los emisores y receptores. Cada grupo multicast tiene un *core* asociado, el cual se usa como punto focal para todos los mensajes de señalización del grupo. Este trabajo deja abiertas investigaciones referentes a la gestión de tráfico y a la entrega fiable de tráficos multicasting.

Concluimos esta sección destacando MCMP (Multiparty Conference Management Protocol) [51] que, sin estar pensado específicamente para ATM, es un protocolo de nivel sesión/transporte distribuido *extremo-extremo* y desarrollado para gestión de grupos de aplicaciones de conferencia. MCMP es un conjunto de algoritmos de control distribuido para configuración de conferencias multipunto y gestión de miembros de grupos de usuarios.

Conceptualmente, MCMP reside en el nivel de sesión en el que se establece la infraestructura para activar la transferencia de información entre los participantes en la conferencia. Pero funcionalmente, el protocolo acompasa los niveles de sesión y de transporte pues utiliza directamente servicios del nivel de red. Son destacables las condiciones de corrección (conectividad, validación, unicidad, consistencia y terminación) que deben ser satisfechas una vez que la conferencia ha sido definida por el algoritmo de configuración de MCMP. El artículo muestra exhaustivas pruebas de corrección para uno de los algoritmos, y también describe la especificación y verificación del protocolo.

2.6. INTEGRACIÓN IP-ATM

Como ya se ha comentado, ATM aporta elevadas prestaciones como son la escalabilidad de su ancho de banda (desde pocos Mbps hasta varios Gbps), su rendimiento (alta capacidad o velocidad) y su habilidad para integrar y soportar datos, audio y video (tráfico multiservicio). Por otro lado, IP ofrece también atractivas características como su simplicidad, su filosofía abierta y el soporte sobre entornos LAN, MAN y WAN.

Estas y otras causas como la creciente expansión de las aplicaciones IP y la actual implantación y uso de redes ATM ha impulsado el rápido crecimiento de ambas tecnologías y en la actualidad existen diversos intentos por combinar las ventajas de ATM e IP para ejecutar aplicaciones Internet sobre redes de tecnología ATM. Esta migración permite a los protocolos y aplicaciones IP disfrutar de las elevadas prestaciones de ATM. Pero la emulación de IP sobre ATM genera comportamientos no óptimos debidos a las diferencias fundamentales entre ambas tecnologías como, por ejemplo, que mientras ATM es una tecnología orientada a la conexión, IP es un protocolo sin conexión. Además, IP tiene problemas para soportar aplicaciones multimedia y/o en tiempo real, mientras éstas son la principal característica para ATM. Por otro lado, ATM tiene problemas para soportar servicios multipunto, mientras IP tiene resueltas sus posibilidades multicast.

En la literatura existen diversas investigaciones para implementar el protocolo IP sobre redes ATM como: classical IP over ATM, LANEmulation, IP switching, Tag switching, Address resolution NARP, next hop resolution NHRP, MPOA y MPLS.

El principal problema para integrar IP y ATM es aprovechar la velocidad y rendimiento de la tecnología de conmutación y también la escalabilidad y flexibilidad de IP por no ser orientado a conexión. Además, las diferencias comentadas entre ambas tecnologías causan duplicaciones de funcionalidad (IP y ATM requieren sus propios protocolos de routing y funciones de mantenimiento y gestión).

El RFC 1577 del IETF Classical IP over ATM es un intento por ejecutar TCP/IP sobre ATM usando el entorno conmutado ATM como enlace de datos para IP [10]. Se plantea el problema de que mientras IPoverATM ofrece una plataforma para comunicación de datos sobre redes heterogéneas, las aplicaciones IP no se pueden beneficiar de las características de QoS de ATM porque la naturaleza de IPoverATM “esconde” las capas por debajo de ATM para poder conseguir el acceso transparente para IP. Algunos de los principales problemas para integrar IP sobre ATM residen en la implementación de IP multicast sobre ATM. Las referencias [10-13] son una excelente literatura en el campo de IP multicast over ATM. Además, la referencia [29] presenta una comparación del ATM-multicast nativo con IP-multicast enfatizando el mapeo entre ambos.

IMSS (IP Multicast Shortcut Service) [32] presenta una solución novedosa para IP multicast sobre redes ATM de área extensa. El gran problema es escalar adecuadamente en un entorno ATM amplio, e IMSS supera los problemas de escalabilidad con una mezcla dinámica de servidores multicast y de conexiones directa y completamente malladas. IMSS es un servicio best-effort de IP-multicast sobre ATM.

NHRP (Next Hop Resolution Protocol) ha sido desarrollado con la intención de facilitar inter-LIS (Logical IP Subnets) VC con el objetivo de aprovechar los beneficios potenciales de ATM que no son empleados en los métodos clásicos como IP over ATM. NHRP es un mecanismo de resolución de direcciones que mapea una dirección IP de destino en una dirección ATM de destino. Aunque NHRP supera algunas de las debilidades de IP over ATM, cuenta también con sus propias limitaciones como la imposibilidad para soportar multicast [52].

IP switching y Tag switching son dos técnicas actuales de conmutación propuestas para ofrecer a las redes ATM mecanismos de routing IP. Ambas están basadas en un mecanismo de intercambio de etiquetas, pero sus implementaciones son diferentes. La referencia [53] demuestra que ambas técnicas mejoran el routing IP clásico, aunque entre ellas no se encuentran diferencias espectaculares en cuanto al rendimiento se refiere. A continuación se comentan brevemente estos dos mecanismos de conmutación.

IP switching (ATM bajo IP) es una alternativa para obviar la conexión extremo-extremo de ATM e integrar su hardware directamente con IP preservando su característica nativa de no orientación a conexión. Para ello implementa IP directamente sobre el hardware ATM mientras mantiene el modelo no orientado a la conexión característico de IP. IP switching soporta también IP multicast y, como la conexión extremo-extremo es desechada, no se necesitan la señalización ni el direccionamiento, siendo únicamente necesarios los protocolos estándares de routing IP. Un conmutador IP es idéntico a un conmutador ATM, sin modificación hardware alguna, pero al que se ha eliminado completamente el software residente en el procesador de control bajo AAL-5 (señalización, protocolo de routing, resolución de direcciones, etc). Constan de GSMP (General Switch Management Protocol) que es un protocolo de bajo nivel para controlar y permitir el acceso del conmutador hardware al controlador de acceso del conmutador IP. El controlador del conmutador IP ejecuta software estándar de routing IP con extensiones como IFMP (Ipsilon Flow Management Protocol) que se encarga de asociar los flujos IP con circuitos virtuales ATM. IFMP puede ser visto como un protocolo de señalización, aunque se ejecuta independiente en cada link y simplemente asocia una etiqueta local con un flujo IP.

Mientras otros métodos (IPoverATM y LANE) proponen un modelo de red como medio lógico compartido encima de la red ATM, IP switching propone un modelo de red más natural para ATM basado en

técnicas punto-a-punto en lugar de usar el modelo clásico de nube [54].

IP switching separa cuidadosamente los actos de etiquetar y conmutar un flujo lo que le permite asegurar la escalabilidad en redes extensas. El etiquetado o la conmutación para un link particular no afecta al resto de las redes.

CSR (Cell Switch Router), como IP Switch, usa un protocolo para ligar un flujo a un VCI y, por tanto, sólo se necesita un único VCI en el controlador del conmutador. La propuesta CSR permite el uso de PVC o SVC en una red ATM. El mecanismo CSR híbrido conmutador/router dispone de todas las funciones habituales en los routers IP y es capaz de ofrecer un servicio de reenvío IP no orientado a la conexión [52].

Por otro lado, la técnica de Tag switching emplea una componente de control y otra de reenvío. El control se encarga de crear y mantener un Tag Information Base (TIB) sobre un grupo de routers tag-switching interconectados. La componente de reenvío usa las entradas existentes en la TIB y la etiqueta de cada paquete para realizar el reenvío de los paquetes a sus correspondientes routers. La información es distribuida entre routers Tag-switched mediante el protocolo TDP (Tag Distribution Protocol).

Para la asignación de una etiqueta a una ruta existen tres esquemas en Tag switching: [53]

- Downstream allocation: esquema llamado así porque las etiquetas salientes son asignadas por los nodos ya atravesados por el flujo de paquetes. El router Tag-switched crea en su TIB una entrada para cada ruta existente en su tabla de rutas. El router asigna una etiqueta a esa ruta y la almacena como etiqueta entrante en su TIB, avisando del hecho a sus routers Tag-switched adyacentes. Cuando las entradas de una ruta han concluido pueden ser usadas por la fase de reenvio.
- El esquema Downstream-on-demand allocation es similar al anterior, salvo en que un router Tag-switched aún no atravesado por el flujo, se encarga de solicitar a los ya pasados una etiqueta para una ruta específica.
- En el esquema Upstream allocation las etiquetas son asignadas por los nodos aún no alcanzados por el flujo, mientras los nodos alcanzados son notificados.

ARIS (Aggregated Route-Based IP Switching) es, como Tag-switching, una aproximación a IPoverATM que proponen asignación de etiquetas en función de la topología de la red. Las etiquetas son asignadas en función de la información obtenida por los protocolos de routing, y las conexiones virtuales son establecidas antes de que el tráfico sea recibido [54]. ARIS introduce el concepto de *egress identifier* para definir granularidad. Para cada valor de identificador de salida el protocolo ARIS establece un árbol multipunto-a-punto [52]. ARIS, como Tag-Switching, soporta multicast.

Aparte de las técnicas de conmutación IP que acaban de comentarse existe también la propuesta oficial del ATM Forum que es MPOA (MultiProtocol Over ATM) basada en otras tecnologías existentes como LANE (que ofrece puenteo de nivel 2) y NHRP (que ofrece encaminamiento de nivel 3). Al combinar puenteo y encaminamiento, MPOA es capaz de soportar protocolos encaminables y no encaminables (IP, IPX, DECnet, AppleTalk, etc.) lo que aporta el calificativo multiprotocolo a MPOA. Aunque no ofrece QoS directamente, ésta es proporcionada por LANE, lo mismo que el soporte multicast. Es importante destacar también que existen investigaciones en marcha para permitir la interoperación entre MPOA y RSVP. MPOA simplifica las comunicaciones entre redes virtuales y, como reduce el protagonismo de los routers en el envío de tráfico, permite obtener mejoras de rendimiento. Los otros métodos de conmutación más importantes (IP Switching y Tag-Switching) quizás aporten mejores prestaciones en cuanto a soporte multicast, QoS y a gestión de tráfico. MPOA ofrece emulación transparente de protocolos enrutados sobre red ATM, lo mismo que LANE ofrece emulación transparente de protocolo LAN sobre red ATM. MPOA opera a niveles 2 y 3 y algunos de sus principales aspiraciones son: permitir a los dispositivos MPOA establecer conexiones ATM directas; integración con LANE; soporte de multicast y broadcast; y separación de conmutación y routing.

Sobre todo cabe destacar que, mientras las propuestas de IETF se preocupan por soportar IP sobre todas las tecnologías estructuradas en capas (ATM es una de ellas), ATM Forum intenta solventar que todos los protocolos de nivel 3 (IP es uno de ellos) se ejecuten sobre ATM.

Existe otra gran familia de soluciones para el soporte de tráfico de datos TCP sobre la clase de servicio ATM-UBR que se encargan de ofrecer rendimiento, justicia y de mantener controlado el retardo. Como veremos en capítulos siguientes, nuestra arquitectura propone en el sistema multiagente un agente programable que se encarga de aplicar este tipo de técnicas a los flujos privilegiados. Comentamos brevemente aquí esta familia de soluciones que no pueden ser considerados como protocolos, para describirlas detalladamente en el Capítulo 5. PPD (Partial Packet Discard), EPD (Early Packet Discard), RED

(Random Early Detection) y FBA (Fair Buffer Allocation) son algunos de los esquemas de gestión de buffers ATM pensados para ofrecer elevado rendimiento y justicia.

EPD es una técnica para mantener la integridad de los paquetes mientras se producen sobrecargas en los conmutadores ATM. Esta técnica de gestión de buffers ATM asegura elevado rendimiento extremo-extremo para aplicaciones que generan datos a ráfagas durante periodos de sobrecarga. En realidad, EPD es uno o varios de los mecanismos propuestos para la gestión de congestiones en redes ATM. La referencia [55] presenta un estudio sobre el comportamiento de EPD ante diferentes dimensiones en los buffers, buscando el punto de compromiso adecuado entre la capacidad de éstos y el correcto comportamiento de EPD.

Por otro lado, PPD es otro conocido esquema de control de congestión para ATM-UBR. PPD descarta las células después de rebasar el tamaño del buffer. La diferencia con EPD es que PPD descarta partes de paquetes. En [56] se observa como PPD alivia el efecto de fragmentación de paquetes, a la vez que presenta una variante de EPD consiguiendo mejorar los parámetros de rendimiento y justicia ofrecidos por este esquema.

RED es otra técnica para mantener elevado rendimiento mientras se minimiza el retardo en la clase de servicio UBR. RED controla el tamaño medio de la cola y se deshace de los paquetes cuando cambia la carga de la red. RED también identifica las conexiones que comparten elevados anchos de banda [57].

El protocolo AREQUIPA (Application REQuested IP over ATM) [58] es un mecanismo que permite a las aplicaciones IP solicitar SVC con QoS garantizada. El RFC 2170 describe este protocolo como un mecanismo para establecer conexiones ATM extremo-a-extremo.

Hay que destacar otro planteamiento general en cuanto a la integración de IP con ATM. En este caso se trata del protocolo RSVP (Resource ReSerVation Protocol) pensado para ofrecer QoS a los flujos IP o IPQoS. IP ofrece entrega best-effort de datagramas que es suficiente para servicios clásicos como correo electrónico, WWW o transferencias de correo. Sin embargo, las aplicaciones multimedia requieren las garantías de servicio que IP no ofrece, por lo que IETF ha creado el grupo de trabajo Integrated Services cuyo principal objetivo es el soporte eficiente en Internet de aplicaciones que requieran garantías de servicio. En este contexto se ha desarrollado RSVP como un protocolo de señalización para Internet encargado de potenciar las redes IP para ofrecer QoS extremo-extremo y no relacionado con ATM, aunque sus objetivos sean los mismos. No obstante, existen trabajos en desarrollo pensados para la interrelación de RSVP con ATM.

Otra novedosa propuesta donde la integración de IP y ATM es una aspiración importante, es MPLS (Multi Protocol Label Switching) [69]. MPLS propone el reenvío de paquetes basado en etiquetas que son asignadas cuando los paquetes entran en la red. Los nodos MPLS reenvían los paquetes y/o células ATM basándose en las etiquetas, en lugar de fijarse en la información IP. Esto permite el reenvío del tráfico de la misma forma, tanto en los conmutadores ATM como en los routers. Sin embargo, el encolado ATM viene dado por los valores de las etiquetas (VCI), y el encolado en los routers está determinado por los valores de determinados bits de la cabecera. Es importante destacar que los conmutadores ATM no pueden analizar las cabeceras de la capa 3, y que las etiquetas pueden ser distribuidas por diferentes protocolos como LDP, RSVP, PIM, BGP, etc.

Concluimos este apartado destacando que otro campo abierto a investigaciones futuras es la integración de protocolos ATM e IP cuando se usan aplicaciones nativas ATM. Sin embargo, no existen demasiadas propuestas en este campo [3].

2.7. NUEVAS LÍNEAS DE INVESTIGACIÓN

En todas las redes existen gran variedad de elementos hardware (switchs, routers, bridges, brouters, hubs, ETD, etc.) que realizan muy diversas funciones (conmutación, routing, puenteo, controles de congestión y de flujo, garantía de QoS, ejecución de aplicaciones, etc.). En la actualidad la red es mayormente un canal de comunicación para transferir paquetes entre equipos finales (ayudada por los elementos hardware citados). Pero también se están realizando importante esfuerzos para equipar a los elementos hardware con elevadas prestaciones aportadas por diversas técnicas software. Esto dota a la red de características activas (active networks) en el sentido que los elementos hardware que la componen computan, modifican u operan los contenidos de los paquetes y también serán capaces de transferir o propagar código. Por tanto, una red activa es una red programable que permite que el código sea cargado dinámicamente en tiempo de ejecución en los nodos de la red.

Las redes activas, abiertas y programables son un nuevo área técnica para explorar vías por las que los elementos de la red puedan ser dinámicamente programados por los administradores de la red, operadores de

red o por usuarios generales con la intención de conseguir QoS y otras características como servicios personalizados. Todo esto aporta atractivas ventajas, pero también importantes cambios en aspectos como el rendimiento, la seguridad y la fiabilidad. Por tanto, esta es una importante línea abierta para investigar y desarrollar técnicas de enrutamiento y protocolos que permitan el movimiento del código, situado en la capa de transporte, hasta los nodos de conmutación de la red.

La literatura estudia varios mecanismos en este campo para obtener ventaja de los nodos activos. El artículo [59] es una excelente revisión de este interesante campo de investigación que discute dos planteamientos en la realización de redes activas: la idea del conmutador programable y el concepto de cápsula. También [60-64] son interesantes referencias de la literatura existente en este tema. Una red es activa si en sus árboles de distribución multicast existen nodos activos con capacidad para ejecutar programas e implementar mecanismos de propagación de código. Algunas de las ventajas de los protocolos activos son conseguidas instalando nodos activos en puntos estratégicos de la red.

Otro nuevo concepto es presentado en [65] como una metodología para el diseño de protocolos. Los *protocol boosters* son una nueva contribución a las redes activas o programables. Una ventaja de los boosters es que pueden ser fácilmente "inyectados" en los sistemas actuales sin provocar cambios en la infraestructura de red.

Por otro lado, [66] propone el concepto de agente de comunicaciones para servicios de comunicaciones multimedia en redes de área extensa. Este artículo introduce una arquitectura software orientada a agente y propone el concepto de agente de comunicación para servicio de comunicación multimedia. Los servicios multimedia son expresados como agentes.

Los conceptos como redes activas, protocolos boosters o agentes software han sido propuestos y desarrollados para redes IP, sin embargo, la actividad empieza a notarse en las redes ATM, y la referencia [67] es una de esas recientes investigaciones. Este artículo muestra agentes software móviles usados para implementar operaciones robustas y funciones de mantenimiento en redes ATM. Los agentes desempeñan un rol similar al de las células OAM en ATM estándar, pues son transmitidos entre entidades de control a intervalos regulares usando recursos predefinidos. La diferencia entre los agentes móviles y las células OAM reside en que éstos pueden contener código.

Por último, destacamos que el IEEE ha propuesto el modelo de referencia IEEE P1520 [68] como el estándar para los interfaces de red programables, que están siendo aplicado a redes ATM, IP y SS7.

En este contexto de las redes activas y agentes software es donde hemos centrado parte de las investigaciones de esta tesis, por lo que dedicamos específicamente el *Capítulo 6* a presentar los aspectos técnicos relacionados con este interesante campo, así como nuestras aportaciones.

2.8. TAP COMO ARQUITECTURA PARA EL SOPORTE DE PROTOCOLO ACTIVO Y NATIVO ATM

Una vez vista la taxonomía de arquitecturas y protocolos ATM, queremos destacar que nuestra propuesta está pensada para soportar los estándares ATM, de forma que se cumple el soporte de las características nativas de esta tecnología. A lo largo de los siguientes capítulos iremos describiendo las diferentes partes de la arquitectura TAP, sin embargo deseamos adelantar en este apartado que hemos realizado su diseño para constituir una Virtual Private Network (VPN) como la mostrada en la *Figura 2.6*. Tal como podemos observar, nuestra pila de protocolos respeta el modelo nativo ATM y únicamente modificamos la capa de adaptación ATM (AAL), donde proponemos una ampliación de AAL-5 en los nodos emisor y receptor. En el caso de los nodos de conmutación de la propia red, únicamente deberán soportar el mecanismo de numeración de las PDU que permita las retransmisiones cuando aparecen las congestiones. La *Figura 2.6* presenta la retransmisión entre dos nodos activos de la arquitectura, pero queremos destacar que TAP no obliga a que la VPN constituida tenga que contener únicamente conmutadores activos que soporten el protocolo TAP, sino que la red puede estar formada por nodos activos y por otros que no tengan por qué soportar TAP. De este modo evitamos la dependencia de la propia arquitectura lo que permite el uso de todo tipo de conmutadores y no sólo nuestros conmutadores activos AcTMs (Active ATM switches).

Los nodos no activos de la VPN actúan como conmutadores normales y, en el caso de las retransmisiones, no realizan ninguna labor especial, salvo el reenvío de células RM en sentido contrario al flujo de datos como si de tráfico ABR se tratase. Esta funcionalidad es posible gracias al soporte de las características nativas del tráfico ATM en la arquitectura propuesta.

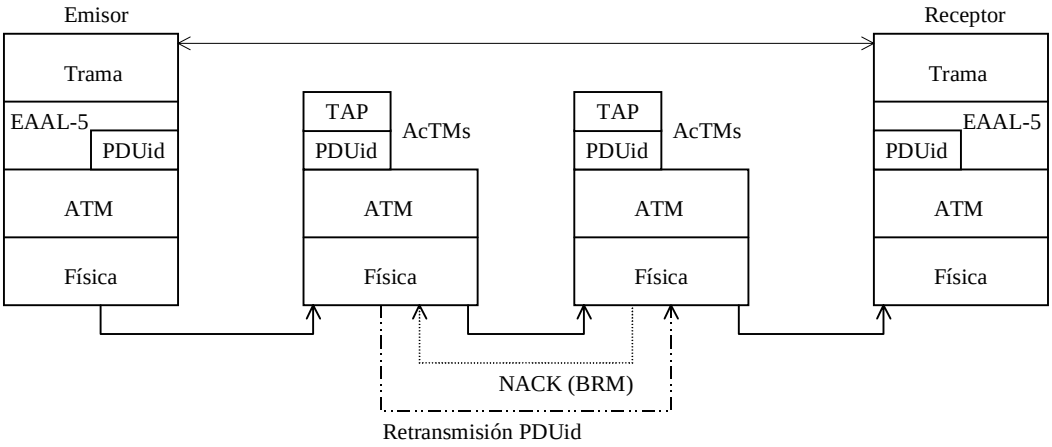

Figura 2.6. *Red Privada Virtual y pilas de protocolos con TAP*

2.9. CONCLUSIONES

En este capítulo hemos descrito algunas de las arquitecturas y protocolos que nos han servido como referencia para el diseño de TAP. Los protocolos nativos pensados para soportar la tecnología ATM son uno de nuestros objetivos [71-73] principales, para que TAP pueda ser extendido a la red con la capacidad de coexistir con las infraestructuras actuales. Para que esto sea posible se han respetado las recomendaciones estándares lo que permite constituir VPN formadas por conmutadores que soportan el protocolo propuesto.

REFERENCIAS

[1] ____ "Native ATM Service: Semantic *Description Version 1," ATM Forum Technical Committee*, ATM Forum Document af-saa-0048.000, (Feb. 1996).

[2] R. Ahuja, S. Keshav and H. Saran, "Design, Implementation, and Performance of a Native Mode ATM Transport Layer," *INFOCOM'96*, Vol. 1 pp. 206-214, (1996).

[3] T. Zahariadis, J. Sanchez-P, C. Georgopoulos, V. Nellas, T. Arvanitis, D. Economou, G. Stassinopoulos, "Native ATM Protocol Stack for Internet Applications in Residential Broadband Networks," *Multimedia Applications Services and Techniques ECMAST'98, Springer*, (May 1998).

[4] R. Venkateswaran, C.S. Raghavendra, X. Chen and V.P. Kumar, "A Scalable, Dynamic Multicast Routing Algorithm in ATM Networks," *IEEE Proceedings INFOCOM'99*, pp 1361-1365 (1.999).

[5] Steinmetz, R.,and Wolf, L.C.,"Quality of Service: Where are We?," *IWQOS'97*, pp.211-221, (1997).

[6] L. Zhang, S. Deering, D. Strin, S. Shenker, D. Zappela, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, pp. 8-18, (Sep. 1993).

[7] ____ "ATM User Network Interface Specification V3.0", *Prentice-Hall, Inc.*, pp. 189-193, (1993).

[8] F. Liaw, D. Perkins, "UNI Procedures and ATM Group Addresses," *ATM Forum Technical Committee, ATM Forum draft document #94-0685R1*, (Sept. 26, 1994).

[9] R. Bubenik, P. Flugstad, "Leaf Initiated Join Extensions," *ATM Forum draft document #94-0325R1*, (July 18, 1994).

[10] G. Armitage, "IP Multicast over UNI 3.0 based ATM Networks," *IETF IP over ATM Working Group Draft*, (Oct 3, 1994).

[11] G. Armitage, "Support for Multicast over UNI 3.0/3.1 based ATM Networks," *RFC 2022. Bellcore*, (Nov, 1996).

[12] G. Armitage, "IP Multicasting over ATM Networks," *IEEE Journal on Selected Areas in Communications. Vol 15, Nº 3*, pp. 445-457 (April, 1997).

[13] G. Armitage, "Multicast and Multiprotocol support for ATM based Internets," *ACM SIGCOMM Computer Communication Review*, pp. 34-46 ().

[14] E. Gauthier, J. Le Boudec and P. Oechslin, "SMART: A many-to-many Multicast protocol for ATM," *IEEE Journal on Selected Areas in Communications. Vol. Nº 3* (April 1.997).

[15] E. Gauthier, J. Le Boudec and P. Oechslin, "Shared Access to Many-to Many ATM Connections," *IEEE Global*

Telecommunications Conference, 1996. GLOBECOM'96. 'Communications: The Key to Global Prosperity'. Volume 3, pp. 2123-2127, (1996).

[16] W. D. Zhong, K. Yukimatsu, "Design requeriments and architectures for multicast ATM switching," *IEICE Trans. Com., Vol E77-B*, pp. 1420-1428, (Nov. 1994).

[17] ____ "ITU Rec. I.150, B-ISDN ATM functional characteristics", *ITU-TS*, (1993).

[18] ____ "ATM user-network interface version 3.1 specification", *ATM Forum*, (1994).

[19] Rec. I.363.5, "Capa de adaptación del modo de transferencia asíncrono tipo 5," *UIT-T*, (08/1996).

[20] ____ "ITU Rec. I.371", *ITU-TS*, (Aug. 1996).

[21] S. Van Luinen, Z. Budrikis, and A. Cantoni, "The Controlled Cell Transfer Capability", *ACM SIGCOMM Computer Communication Review*, pp. 55-71, (1997).

[22] ____ "Traffic Management Specification Version 4.0," *ATM Forum Technical Committee, ATM Forum Document af-tm-0056.000*, (April 1996).

[23] M. de Prycker, "Asynchronous Transfer Mode. Solution for Broadband ISDN (3ª Ed.)," *Ed. Prentice Hall*,(1995)

[24] ____ "ATM Name System Specification", Version 1.0, *ATM Forum*, (Nov. 1996).

[25] D. Kandlur, D. Saha and W. Willebeck, "Protocol Architecture for multimedia Application over ATM Networks," *IEEE Journal Selected and Communications Vol. 14, Nº 7*, pp. 1.349-1.359, (Sep. 1996).

[26] R. Ahuja, S. Keshav and H. Saran, "Design, Implementation, and Performance Measurement of a Native-Mode ATM Transport Layer (Estended Version)," *IEEE/ACM Transactions on Networking*, Vol 4, Nº 4, (August 1996).

[27] R. Karabek, "A Native ATM Protocol Architecture Design and Performance Evaluation," *IEEE Proceedings 22nd Annual Conference on Local Computer Networks*, pp. 204-210, (1997).

[28] R. Karabek, M. Scholz, "A Transport Protocol for Native Mode ATM Networks Design and Implementation," *EEE GLOBECOM'96 Workshop:Transport Protocols for High Speed Broadband Networks*, London, (November 1996).

[29] R. Bagwell, J. McDearman and D. Marlow, "A Comparison of Native ATM-Multicast to IP-Multicast With Emphasis On Mapping Between the Two," *IEEE, Proceedings og the Twenty-Seventh Southeastern Symposium on System Theory*, (1995).

[30] Saran, "The IDLInet Native Mode ATM Protocol Stack," *Indian Institute Technology, Delhi, http://cell-relay.indiana.edu/cell-relay/publications/software/IDLInet.*

[31] D. C. Feldmeier "A framework of architectural concepts for high-speed communications systems," *IEEE J. Select. Areas Commun.*, Vol. 11, Nº 4, (May 1993).

[32] T. Anker, D. Breitgand, D. Dolev, Z. Levy, "IMSS: IP Multicast Shortcut Service," *INFOCOM'98*, (1998).

[33] T. Anker, D. Breitgand, D. Dolev, Z. Levy, "Congress: CONnection-oriented Group-address RESolution Service," *Proceeding of SPIE'97* vol 3233 on Broadband Networking Technologies, Nov. (1997), *http://www.cs.huji.sc.il/transis/*

[34] *kStack http://comet.columbia.edu/software/kStack*

[35] T. La Porta and M. Schwartz, "Architectures, Features, and Implementation of High-Speed Transport Protocols," *IEEE Network Magazine*, pp. 14-22, (May. 1991).

[36] T. Matsuda, K. Matsuda, "A new protocol processing architecure for high-speed networks," *http://ipdps.eece.unm.edu/1998/* (1998).

[37] ____ "B-ISDN, ATM Adaptation Layer Service. Specific Connection Oriented Protocol (SSCOP) Q.2110," *ITU-T*, (10-Mar. 1994).

[38] J. Solé-Pareta, J. Vila-Sallent, "Network-based paralell computing over ATM using improved SSCOP protocol," *Computer Communications* 19 pp. 915-926, (1996).

[39] P. Goyal, Vin, H., Shen, C., Shenoy, P., "A reliable, Adaptive Network Protocol for Video Transport", *INFOCOM'96*, Vol 3, pp. 1080-1090, (1996).

[40] K. Obraczka, "Multicast Transport Protocols: A survey and Taxonomy," *IEEE Comm. Magazine*, (1998).

[41] P. E. Boyer and D.P. Tranchier, "A reservation principle with applications to the ATM traffic control," *Computer Networks ISDN Systems*, vol. 24, (1992).

[42] S. Manthorpe, Jean-Yves Le Boudec, "A comparison of ABR and UBR to support TCP traffic," *Technical report Laboratoire de Réseaux de Communication, EPFL http://lrcwww.epfl.ch/*, (1997).

[43] Woo-Seop Rhee, Yoon-Young An, Hwa-Suk Kim, Hong-Shik, "Interoperability Mechanism of ABR/ABT Capability in ATM Public Networks," *IEEE ICC'97 Montreal, Towards the Knowledge Millenium. 1997 IEEE International Conference on Communications* Volume 3 pp. 745-749, (1997).

[44] You-Ze Cho, Myeong-Yong Lee, "An Efficient Multicast Algorithm for ABR Service in ATM Networks," *IEEE ATM Workshop 1997. Proceedings pp. 351-360*, (1997).

[45] Kerry W. Fendick, "Evolution of Controls for the Available Bit Rate Service," *IEEE Communications Magazine*, pp. 35-39, (Nov. 1996).

[46] S. Fahmy, R. Jain, R. Goyal, B. Vandalore and S. Kalyanaraman, "Feedback Consolidation Algorithms for ABR Point-to-Multipoint Connections in ATM Networks," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Volume 3 pp.* 1004-1013, (1998).

[47] Lotfi Benmohamed, Y. T. Wang, "A Control-Theoretic ABR Explicit Rate Algorithm for ATM Switches with Per-VC Queueing," *INFOCOM'98, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 1*, pp. 183-191, (1998).

[48] Sharat Prasad, Kamran Kiasaleh and Poras Balsara, "LAPLUS: An Efficient, Effective and Stable Switch Algorithm for Flow Control of the Available Bit Rate ATM Service," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 1* pp. 174-182, (1998).

[49] Eitan Altman, Tamer Basar and R. Srikant, "Robust Rate Control for ABR Sources," *IEEE Proceedings INFOCOM'98*, pp. 166-175, (1998).

[50] Matthias Grossglauser and K.K. Ramakrishnan, "SEAM: Scalable and Efficient ATM Multicast," *IEEE Proceedings INFOCOM'97*, pp. 867-875, (1.997).

[51] M. Nguyen, and M. Schwartz, "MCMP: A Transport/session level Distributed Protocol for Desktop Conference Setup," *IEEE Journal Selected and comunications, Vol. 14 Nº 7*, pp. 1404-1421, (Sept. 1996)

[52] Paul Patrick White, "ATM Switching and IP Routing Integration: The Next Stage in Internet Evolution," *IEEE Communications Magazine*, April 1.998, pp. 79-83 (1.998).

[53] Xipeng Xiao, Lionel M. Ni and Vibhavasu Vuppala, "A comprehensive comparison of IP Switching and Tag Switching," *IEEE Proceedings ICPADS'97*, pp. 669-675 (1.997).

[54] Peter Newman, Greg Minshall and Thomas L. Lyon, "IP Switching-ATM Under IP," *IEEE/ACM Transactions on networking, Vol. 6 Nº 2*, April 1.998 pp. 117-129 (1.998).

[55] Maurizio Casoni and Jonathan S. Turner, "On the Performance of Early Packet Discard," *IEEE Journal On Selected Areas in Communications, Vol. 15, Nº 5*, June 1.997, pp. 892-902, (1.997).

[56] Kangsik Cheon and Shivendra S. Panwar, "Early Selective Packet Discard for Alternating Resource Access of TCP over ATM-UBR," *IEEE Proceedings LCN'97*, 1.997, pp. 306-316, (1.997).

[57] Omar Elloumi and Hossam Afifi, " RED Algorithm in ATM Networks," *IEEE Proc. ATM workshop*, pp. 312-319, (1.997).

[58] Almesberger, J. Le Boudec and Ph. Oechslin, "Arequipa: TCP/IP over ATM with QoS for impatient," *http://lrcwww.epfl.ch/WebOverATM/about_woa/arequipa.html*

[59] D. Tennenhouse et al., "A Survey of Active Network Research," *IEEE Communications Magazine*, (1997).

[60] U. Legedza, D. Wetherall and J. Guttag, "Improving the performance of Distributed Applications Using Active Networks," *INFOCOM'98*, pp. 590-599, (1998).

[61] D. Tennenhouse, D. Wetherall, "Towards an active network architecture," *Multimedia Computing and Networking 96*, San Jose, CA, (Jan 1996).

[62] Li-wei H. Lehman, Stephen J. Garland and David L. Tennenhouse, "Active Reliable Multicast," *INFOCOM'98 pp. 581-589*, (1.998).

[63] Rosen Sharma, S. Keshav, Michael Wu and Linda Wu, "Environments for Active Networks," *IEEE Proceedings INFOCOM'99*, pp. 77-84, (1.999).

[64] David J. Wetherall, John V. Guttag and David L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols," *IEEE Proceedings OPENARCH'98*, pp. 117-129, (1.998).

[65] David C. Feldmeier, Anthony J. McAuley. Jonathan M. Smith, Deborah S. Bakin, William S. Marcus and Thomas M. Releigh, "Protocol Boosters," *IEEE JSAC* Vol. 16, Nº 3, pp. 437-443, (April 1.998).

[66] R. Kishimoto, "Agent communication system for multimedia communication services," *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking the Next Generation., Proceedings IEEE Vol. 1*, pp. 10-17, (1996).

[67] David A. Halls and Sean G. Rooney, "Controlling the Tempest: Adaptive Management in Advanced ATM Control Architecture," *IEEE JSAC* Vol. 16, Nº 3, pp. 414-423, (April 1.998).

[68] P1520 - Proposed IEEE Standard for Application Programming Interfaces for Networks, *http://www.iss.nus.sg/IEEEPIN/*

[69] A. Viswanathan, "Evolution of Multiprotocol Label Switching," *IEEE Communications Magazine*, pp. 165-173, (1998).

[70] Josep Mangues-Bafalluy and Jordi Domingo-Pascual, "Multicast Forwarding over ATM: Native approaches," *IEEE Communication Surveys & Tutorials http://www.comsoc.org/pubs/surveys/*, Third quarter 2000 (2000).

[71] José Luis González-Sánchez y Jordi Domingo-Pascual, “Revisión y clasificación de protocolos para redes ATM”, *Jornadas Técnicas de RedIRIS’98*, (1998).

[72] José Luis González-Sánchez and Jordi Domingo-Pascual, “Protocols for ATM Networks: Survey, Classification and New Trends” *TR UPC-DAC-1999-65. http://www..ac.upc.es/recerca/reports/INDEX1999DAC.html* (1.999).

[73] José Luis González-Sánchez and Jordi Domingo-Pascual, “Protocols for ATM Networks: Survey, classification and new trends,” *IX Jornadas TELECOM I+D’99*, (1999).

CAPÍTULO 3

FIABILIDAD Y GARANTÍA DE SERVICIO

3.1. INTRODUCCIÓN

En los sistemas de comunicación se considera que un bit es erróneo cuando llega a su destino con un valor distinto del que tenía al ser transmitido. Según esto, se expresa el parámetro BER (Bit Error Rate) como el ratio entre los bits erróneos y los bits transmitidos en un espacio de tiempo representativo, y se considera como el parámetro más importante para caracterizar las imperfecciones en los sistemas de comunicación digitales. Como los bits son agrupados para su transmisión en paquetes (células en el caso ATM) pueden provocarse grupos de errores por pérdidas de paquetes o por un enrutamiento incorrecto de éstos. En los sistemas de transmisión suelen producirse errores de bits individuales, mientras en los sistemas de conmutación y multiplexación los errores pueden ser de bits y/o paquetes. En [6] se presentan los planteamientos teóricos y los diferentes ratios a considerar debidos a los errores producidos en las transmisiones.

La tecnología ATM se caracteriza por su excelente comportamiento ante diversos tipos de tráfico y por ofrecer la posibilidad de negociar los parámetros generales de QoS [1] como el *throughput*, *delay*, *jitter* y la *reliability*. La fiabilidad (*reliability*) es, quizás, el parámetro menos estudiado de los cuatro y es aquí donde encontramos la motivación principal de este capítulo.

Como hemos visto en el Capítulo 1, la unidad básica de conmutación y multiplexación en las redes ATM es la célula de tamaño fijo de 53 bytes. Sabemos [2] que la cabecera tiene, entre otros campos, el HEC (Header Error Control) de 8 bits, usado como CRC (Cyclic Redundancy Code) únicamente para el control de errores de cabeceras. La tecnología no dispone de ningún mecanismo de control de errores para el campo de datos de las células unitarias, aunque sí existen propuestas de aplicación de CRC para células agrupadas en paquetes o PDU (Protocol Data Unit) que son la unidad de procesamiento de la capa AAL. Es importante destacar que en ATM el control de errores se realiza extremo-extremo por los terminales de la comunicación, lo que afecta negativamente al rendimiento de la red de muy diversas formas como que, por ejemplo, la pérdida de una única célula provoca errores de reensamblado de CRC en la capa AAL-5, lo que requiere la retransmisión extremo-extremo de una PDU completa.

Las redes ATM pueden experimentar tres tipos de errores [2-5]: bits erróneos que corrompen una porción de datos de una célula; errores de conmutación debidos a errores no detectados en las cabeceras de las células y bits perdidos debidos a congestiones. Las investigaciones realizadas demuestran que en ATM las pérdidas por congestión son la forma predominante de errores dado que los elevados márgenes de fiabilidad aportados por la fibra óptica como medio de transmisión principal (probabilidades de error entre 10^{-8} y 10^{-12}) evitan la aparición de los otros dos tipos de errores citados. Los errores debidos a bits erróneos son menos frecuentes y -en bastantes casos- menos importantes, ya que muchas aplicaciones de tiempo real prefieren perder células antes que soportar el retardo provocado por la retransmisión de células erróneas o congestionadas.

En el caso de los errores de conmutación y/o multiplexación los paquetes erróneos son causados por errores en las cabeceras que provocan errores de interpretación o de enrutamiento y acaban provocando la pérdida de los paquetes o su llegada a un destino equivocado. El propio dispositivo de conmutación o multiplexación también puede desechar células ATM por agotamiento de sus recursos cuando muchas células

compiten por ellos, lo que acaba generando también errores. La unidad de pérdida es la célula ATM, y éstas pueden ocurrir a ráfagas afectando a varias células consecutivas en una misma conexión.

La teoría de codificación de información distingue entre dos tipos de corrupción de datos: un error se define como un bit con valor desconocido en una posición desconocida; y una borradura (*erasure*) es un bit con un valor desconocido en una posición conocida. Con este planteamiento, es destacable que los métodos de codificación aspiran a convertir o reemplazar los errores por *erasure* que son más fácilmente tratables, con lo que aumenta la eficiencia de los códigos correctores como FEC (Forward Error Correction).

En muchos casos suele recurrirse a protocolos de la capa de transporte extremo-extremo para incrementar la fiabilidad mediante retransmisiones o usando códigos generadores de información redundante. La literatura [6] describe tres técnicas básicas para ofrecer fiabilidad: Automatic Repeat Request ARQ [7,21], Forward Error Correction FEC [8-12] y también mecanismos híbridos de ARQ combinados con FEC. El objetivo de este capítulo es centrar estos mecanismos de fiabilidad en ATM para realizar un análisis comparativo que nos permita elegir el más adecuado para aportar garantía a las transferencias privilegiadas objeto de la arquitectura TAP.

En primer lugar describiremos las diferencias entre el parámetro de QoS fiabilidad y el de Garantía de Servicio como concepto propuesto en nuestra tesis para aportar una nueva característica menos laxa, aunque no menos importante, que la propia fiabilidad. Pasaremos después a estudiar los tres citados mecanismos de fiabilidad, para centrarnos luego en la técnica de retransmisión que aplicamos en TAP para ofrecer la Garantía de Servicio a las conexiones privilegiadas.

3.2. FIABILIDAD Y GARANTÍA DE SERVICIO

En comunicaciones el concepto fiabilidad está claramente aceptado como la forma de aportar a las conexiones extremo-extremo garantía plena de que la información que transfieren llega sin ningún error o, si aparecen errores, todos pueden ser detectados y corregidos. Aplicar a una red, a un protocolo, o a una tecnología de comunicaciones el calificativo de fiable implica que ésta puede aportar la garantía de transferencias libres de errores. Para conseguir la fiabilidad total existen dos posibilidades: una consistente en aplicar un mecanismo de control en el que todos los tipos de datos transferidos son confirmados por el destino, con la intención de garantizar que no se pierde ni una sola unidad de transferencia; y la otra consistente en añadir información redundante a los paquetes de información que garantice a los receptores que esos paquetes no han sufrido ninguna variación en la red. En realidad, la fiabilidad se consigue uniendo estas dos técnicas, ya que la primera garantiza que no hay pérdidas de datos, y la segunda que los datos son correctos. Cuando se detectan pérdidas y/o errores se recurre a la retransmisión extremo-extremo entre el emisor y el receptor. Podemos ver cómo el concepto de fiabilidad no se enfrenta directamente a los problemas provocados por las congestiones, aunque veremos más adelante cómo existen pesadas técnicas de confirmación que pueden ser usadas para detectar las congestiones, aunque sea a costa de pérdida del *goodput*[1] en la red.

En nuestro caso, no pretendemos centrarnos plenamente en el ámbito de la fiabilidad, sino en encontrar un mecanismo que aporte a las transferencias ATM su garantía del servicio en el caso que aparezcan congestiones en los conmutadores y, además que, cuando esto se produce, pueda ser solventado de forma local a los conmutadores congestionados sin implicar a toda la red para optimizar el goodput. Para ello proponemos un protocolo que no calificamos de fiable ya que en el sentido estricto de esta palabra no se comporta como tal, sino que lo que aporta es un elevado grado de confianza de servicio a las conexiones que lo utilizan. Así, nuestra propuesta es identificada con el nombre TAP (Trusted and Active Protocol) y, antes de pasar a describir sus características, deseamos justificar el adjetivo *trusted* del nombre de nuestro protocolo.

En primer lugar, queremos destacar que no existe consenso en la literatura en cuanto a lo que es, o lo que representa el término *trust*, no obstante, muchos investigadores han reconocido su importancia. Las investigaciones que nos afectan usan la definición de *trust* en una línea muy específica relacionándola con términos como autentificación y fiabilidad en el ámbito de las comunicaciones [13,14]. Sin embargo, otros autores entienden la palabra *trust* de una forma mucho más genérica relacionándola con aspectos de la personalidad humana, la sociología, la economía e incluso la psicología.

El diccionario Webster define *trust* con las siguientes acepciones:

[1] El concepto goodput expresa el rendimiento de la red considerando no sólo la capacidad de transferencia (throughput), sino también el efecto generado por las retransmisiones.

- *An assumed reliance on some person or thing. A confident dependence on the character, ability, strength or truth of someone or something.*
- *A charge or duty imposed in faith or confidence or as a condition of a relationship.*
- *To place confidence (in an entity).*

La mayor parte de autores destacan las implicaciones de estas definiciones y combinan sus resultados con la perspectiva social de *trust* para crear la definición de *trust* en un sistema " *a belief that is influenced by the individual's opinion about certain system features*" [15]. No obstante, podemos implicar en el concepto a las entidades que deben participar de la característica *trusted* si nos fijamos en la definición que hace de la palabra *trust* el Oxford Reference Dictionary "*the firm belief in the reliability or truth or strength of a entity*". De este modo, una entidad *trusted* tendrá una elevada fiabilidad y no deberá fallar en el transcurso de una interacción, que realizará un servicio o acción en un periodo razonable de tiempo. En nuestro trabajo, tratamos el término *trust* en el contexto de las redes ATM y de los sistemas de computación distribuida, en los cuales no sólo deben ser garantizados los extremos de la comunicación, sino también los conmutadores que los interconectan y las interacciones ofrecidas por los servicios que soportan. Como podemos ver, *trust* es realmente una composición de muy diferentes atributos como fiabilidad, dependencia, honestidad, seguridad, fortaleza, confianza o veracidad, los cuáles deben considerarse como dependientes del entorno en los cuáles sea definida la palabra *trust*.

Por tanto, debemos remarcar las diferencias conceptuales entre las palabras *trusted* y *reliability*. Como hemos destacado antes, en el contexto de las redes de comunicación la palabra *reliable* se ha venido usando de forma ampliamente aceptada para identificar las transmisiones plenamente garantizadas por diferentes mecanismos como las técnicas FEC descritas en este capítulo. Nuestro objetivo no es la búsqueda de esa fiabilidad completa en las conexiones, que es ofrecida en ATM por muy diversos métodos y siempre a costa de sobredimensionar las cabeceras de las células o PDU ATM, y con retransmisiones extremo-extremo que, por otro lado, no solventan los problemas debidos a las congestiones en los conmutadores. Por tanto, ya que el concepto *reliable* está ampliamente consensuado para aportar fiabilidad extrema, hemos decidido usar el término *trust* para destacar que nuestro objetivo es ofrecer Garantía de Servicio a fuentes de tráfico, generalmente ABR y UBR, que se desean fiables cuando aparecen congestiones en la red.

Aclaradas estas diferencias destacamos que, en torno al protocolo TAP, aparece la idea de transmisiones garantizadas, lo que nos ha dado pie para aportar el concepto de Garantía de Servicio (que identificamos en lo sucesivo como GoS) a las transferencias que se estime que así lo necesitan. Para nosotros la GoS será en realidad un nuevo parámetro de QoS que se deriva directamente del parámetro de fiabilidad. A partir de ahora nos referiremos por tanto al hecho que TAP aporta GoS a aquellas conexiones que se desea que tengan garantía en las transferencias y en este mismo capítulo vemos en las secciones siguientes el fundamento técnico en que nos vamos a apoyar para conseguir la GoS.

3.3. CÓDIGOS DE REDUNDANCIA CÍCLICA (CRC)

Como ha quedado ya indicado, nuestro objetivo no es resolver los errores producidos en las transmisiones, sino aportar una solución a las pérdidas provocadas por la aparición de congestiones. No obstante queremos, por la estrecha relación que existe y dado que para aportar la GoS nos vamos a apoyar en técnicas de recuperación de paquetes, aclarar algunos aspectos relacionados con la detección y corrección de errores.

Las redes ATM, como otras tecnologías, están sometidas a la aparición de errores, tanto aislados como a ráfagas. Son ya conocidos los diversos mecanismos que pueden usarse para enfrentarse a los errores producidos en los medios de transmisión, y todos ellos se basan en la estrategia de incluir más o menos código redundante en la información que se desea transferir. Cuanto mayor fiabilidad se desea aportar a las transmisiones, más información redundante habrá que añadir a los datos originales. Así, se pueden usar códigos redundantes para la detección de errores y códigos redundantes para la corrección de esos errores. Para los primeros se usan los conocidos códigos de Hamming, mientras para los segundos se emplean los, también conocidos, códigos polinómicos o Códigos de Redundancia Cíclica (CRC).

3.3.1. CRC APLICADOS A CABECERAS DE CÉLULAS ATM

En el caso de ATM se han realizado investigaciones en el campo de los CRC para garantizar la corrección, tanto de las cabeceras, como de los campos de datos de las células ATM. Así, como puede observarse en la *Figura 1.1* del Capítulo 1, la cabecera de las células aporta el campo HEC que realiza las

comprobación de corrección de cada uno de los bits de la cabecera, pero no realiza ninguna verificación de los bits del campo de carga útil de las células. El campo HEC tiene una longitud de 8 bits (para aportar fiabilidad a los 32 bits restantes de las cabeceras) con el que se puede conseguir un código redundante suficientemente potente para detectar y corregir todos los errores de un solo bit, y también es capaz de detectar el 90% de los errores de más de un bit.

El polinomio generador de grado 8 usado en el CRC es $x^8 + x^2 + x + 1$, que es más que suficiente para considerar fiables las cabeceras ATM, y máxime cuando se conoce que en la fibra óptica, que es su principal medio físico de transmisión, los estudios realizados demuestran que el 99,64% de los errores son de un solo bit. Sabemos además que la probabilidad de que se produzca un error en un bit enviado a través de fibra es hoy inferior a 10^{-12} . Por tanto, la probabilidad de que aparezca un error de más de un bit está en torno a 10^{-15}. A la vista de estos números podemos asumir que, aunque la fiabilidad no es absoluta, puede decirse que la probabilidad de recibir células con cabeceras erróneas es prácticamente nula.

Mención a parte merece el caso del campo de datos de las células que, como hemos comentado, no es considerado por el campo HEC al aplicar el código polinómico. Esto es así porque poder garantizar también los 48 bytes del campo de datos de la célula requeriría de un campo HEC superior a 8 bits, lo que supondría usar un octeto más en las cabeceras que pasarían a 6 octetos y uno menos en el campo de datos que quedarían con 47 octetos. La solución podría consistir también en replantear el tamaño total de las células de 53 octetos, pero es conocido el esfuerzo que costó encontrar el consenso hasta llegar a este peculiar tamaño de células. En lugar de esto lo que propone la tecnología ATM es el uso de CRC aplicado sobre paquetes de células que engloban también a los campos de datos. Esta labor se realiza en la capa AAL como vamos a comprobar a continuación.

3.3.2. CRC APLICADOS A PAQUETES DE CÉLULAS

Aunque la célula es la unidad de conmutación, multiplexación y transferencia dentro de la red ATM, las aplicaciones pueden usar unidades de transmisión de mayor tamaño y propias de los protocolos de las capas superiores que están sobre la capa AAL (ver *Figura 1.2*). Es sabido que existen varios protocolos en la capa AAL, pensados inicialmente para cada una de las CoS. Estos protocolos son AAL-1, AAL-2, AAL-3/4 y AAL-5. Pues bien, sin entrar en excesivas consideraciones, en este caso nos interesa destacar que, de un modo u otro, las cuatro variantes de AAL aportan un mecanismo de comprobación para aportar fiabilidad a los paquetes de datos que transfieren. Mientras AAL-1 emplea un número de secuencia que hace las veces de suma de comprobación, las otras tres emplean CRCs para garantizar que la información no experimenta errores.

En nuestro caso particular vamos a detenernos en el caso particular de AAL-5 ya que, como adelantamos en el *Capítulo 2*, modificamos y ampliamos este protocolo para conseguir la GoS en la arquitectura TAP.

Desde el punto de vista de un emisor, la capa AAL-5 (ver *Figura 1.2*) pasa información a la capa ATM en forma de ATM-SDU (Service Data Unit) de 48 octetos. Desde el punto de vista del receptor la capa AAL-5 recibe de la capa ATM unidades en forma de ATM-SDU de 48 octetos. En líneas generales, lo que hace el protocolo AAL-5 es tomar secuencias de mensajes del nodo emisor provenientes de los protocolos superiores y aplicar la segmentación a los paquetes de información para conseguir una secuencia de células de 48 octetos, a las que la capa ATM se encargará de añadir la cabecera para poder ponerlas en la red. Para realizar estas funciones el protocolo que reside en la capa AAL-5 emplea una serie de primitivas y usa un conjunto de servicios que están definidos en la Rec. I.363.5 [4].

AAL-5 está dividida en dos subcapas, la subcapa SAR (Segmentation and Reassembly) y encima de ésta, la subcapa CPCS (Common Part Convergence Sublayer). Existe una tercera subcapa que puede ser nula y estar sobre CPCS, que se llama SSCS (Service Specific Convergence Sublayer). Cada una de ellas dispone de sus propias funciones, primitivas y servicios. Nosotros nos hemos fijado en la CPCS para comprobar las posibilidades de fiabilidad y de GoS de esta capa. La *Figura 3.1* muestra el formato de las PDU (Protocol Data Unit) que es la unidad de transferencia en la subcapa de convergencia de la parte común de AAL5. Como podemos observar, las PDU constan de dos partes bien diferenciadas con un campo de datos y una cola.

- El campo de datos es de tamaño variable y puede contener de 1 a 65.535 octetos de información.
- Para que el campo de datos contenga siempre un número exacto de unidades de células y, por tanto sea múltiplo de 48, se emplea un campo de relleno o *padding* que puede tener de 0 a 47 octetos.
- Por su parte, la cola de la CPCS-PDU cuenta con 8 octetos divididos en los siguientes campos:

- ✓ El campo CPCS-UU (User to User), de 1 octeto, se emplea para transferir información transparente entre usuarios de la subcapa CPCS. Este campo en realidad no es usado por la capa AAL y está a disposición de capas superiores como luego veremos.
- ✓ El campo CPI (Indicador de Parte Común), de 1 octeto, se usa para que las colas de PDU queden alineadas o ajustadas a su tamaño total fijo de 64 bits. En realidad este campo, como el anterior, puede ser usado libremente por los protocolos de capas superiores pues el estándar no le ha encomendado funciones específicas.
- ✓ El campo Longitud, de 2 octetos, expresa en binario el tamaño real del campo de datos que, como hemos dicho antes, es de tamaño variable. Por tanto, este campo nos permite delimitar el tamaño real de cada PDU. También es usado por los nodos receptores para determinar si una PDU ha experimentado algún tipo de pérdida o ganancia de información. Es decir, sirve también para detectar errores.
- ✓ El campo CRC, de 4 octetos, es el que realmente se encarga de ofrecer la fiabilidad a todos los bits de la PDU. Este campo se rellena con el resultado de aplicar un CRC de 32 bits y su correspondiente polinomio generador de grado 31 a todo el campo de datos, al relleno y a los primeros 4 octetos de la cola de las PDU. Este último campo es el que acaba aportando la fiabilidad a las PDU individuales y, por tanto, a los campos de datos de las células ATM que no son protegidos por el CRC HEC de las cabeceras de las células.

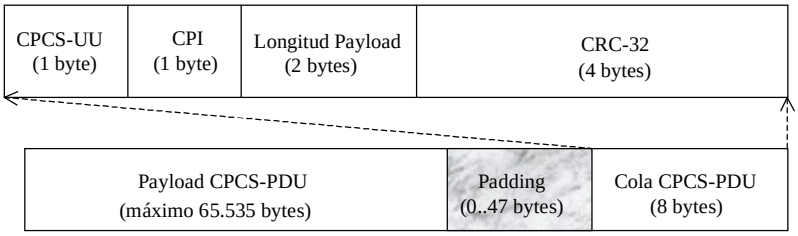

Figura 3.1. *Formato de las PDUs de CPCS de AAL-5*

Hemos podido comprobar cómo, de un modo u otro, tanto las cabeceras como los campos de datos pueden ser protegidos de los inesperados problemas que la red pueda experimentar. Nos encontramos con que el precio que hay que pagar por esta seguridad es de 8 bits para cada una de las células individuales, y de 32 bits en el caso de cada una de las PDU. Este precio se paga por tanto en forma del *overhead* que hay que introducir para enviar el código redundante en cada unidad de transferencia, ya sean células o PDU. Pero otro problema añadido es que los errores son sólo solventables extremo-extremo. Tanto el throughput como el goodput se ven afectados para poder disponer de la fiabilidad aportada por HEC y FEC.

3.4. AUTOMATIC REPEAT REQUEST (ARQ)

ARQ [7] es una técnica de bucle cerrado basada en la retransmisión de datos que no han sido recibidos correctamente debido a los problemas que ya hemos comentado en los puntos anteriores. En realidad, las técnicas ARQ se usan conjuntamente con los CRC ya que, cuando un CRC detecta un error que no es capaz de solventar, necesitará de un mecanismo de solicitud de retransmisión de la trama errónea desde el emisor. ARQ será la técnica que usaremos para notificar al emisor de las retransmisiones que debe realizar.

En el caso de aplicaciones de tiempo no real la retransmisión de información incorrecta o perdida se puede confiar a técnicas ARQ que se han demostrado como poco apropiadas para servicios de tiempo real o con requerimientos de baja latencia por el elevado retardo que introducen las retransmisiones. ARQ ofrece dos variantes y ambas requieren que emisor y receptor intercambien algún tipo de información de estado por lo que incurren en retardos para el nodo receptor, *implosión*[2] en el nodo emisor y excesivo *overhead* en la red.

- En la primera variante de ARQ el receptor devuelve mensajes de confirmación positiva (ACK+) incluso cuando ha recibido correctamente los datos. Este es el mecanismo tradicionalmente usado para aportar fiabilidad en las transmisiones unicast. Para poder implementar este protocolo es necesario

[2] La implosion es el efecto negativo que experimentan las fuentes emisoras de tráfico cuando deben atender las solicitudes de retransmisión de células perdidas o erróneas. En el caso de las transferencias multipunto la implosión genera importantes problemas.

que las tramas que va a procesar tengan un número de secuencia que sirva para identificarlas en la recepción y en el proceso de retransmisión. Este planteamiento responde claramente al modelo de protocolos de la capa de enlace (también de transporte) orientados a conexión y con acuse de recibo que ofrecen fiabilidad a la capa de red. El carácter orientado a conexión elimina, incluso, la posibilidad de tramas repetidas que, en el caso de ser no orientado a conexión, pueden aparecer en los *time-out* de los ACK. En el caso de ATM este es un protocolo que puede aportar fiabilidad, pero nos encontramos, como poco, con tres grandes inconvenientes: el primero -solventable- es el de la no existencia de números de secuencia en las células ATM que impide la posibilidad de retransmisión de células concretas; el segundo -más grave- el de la elevada latencia y degeneración del goodput provocados por tener que confirmar desde el receptor la llegada de cada célula del emisor; y el tercero -inabordable en multicast- el problema de implosión que provoca que los emisores se vean inundados con los acuses de recibo de los receptores. Parece claro que este protocolo de confirmación positiva está pensado para unidades de transferencia mucho más grandes que el tamaño de una célula y que además dispongan de un mecanismo de numeración que permita establecer la secuencia de llegadas y detectar las pérdidas cuando se altera la secuencia.

- En la segunda variante de ARQ el receptor devuelve mensajes de acuse de recibo negativos (NACK) sólo cuando se han producido errores o pérdidas de datos. Es claro que este mecanismo de detección y recuperación de errores solventa o aminora el segundo y tercer problemas comentados en NACK+. En este caso no se sobrecarga al emisor ni a la red con acuses de recibo innecesarios, ya que sólo se generarán cuando el receptor detecta problemas y solicita la retransmisión de la trama perdida o errónea. Desde luego, para poder implementarlo es necesario que las tramas dispongan de un número de secuencia que sirva de referencia para las retransmisiones. Por otro lado, la técnica NACK aporta menor fiabilidad cuando las peticiones de retransmisión se pierden.

Además de las intuitivas ventajas a favor de NACK, investigaciones como [21] demuestran que ACK+ es no escalable en el caso de protocolos multicast ya que a medida que va creciendo el número de receptores el funcionamiento se va degradando. Debemos destacar, no obstante, que exitosos protocolos p-p como TCP, HDLC y TP4 y otros tantos multicast/broadcast usan el planteamiento ACK+ para conseguir la fiabilidad.

3.5. FORWARD ERROR CORRECTION (FEC)

FEC [8-12] es una interesante alternativa a las dos variantes de ARQ pues ofrece fiabilidad sin incrementar la latencia extremo-extremo. Su principio de funcionamiento se basa en la codificación de paquetes en el emisor con información redundante junto con los datos, de forma que sea posible reconstruir los paquetes originales en el receptor, reduciendo, o incluso eliminando, las retransmisiones y el negativo efecto de la *implosión* sobre los emisores. Para que el método sea eficiente, el tamaño de la información redundante debe ser menor que los datos de información que se desea hacer fiables. FEC consiste en el uso de esquemas de codificación compleja que añaden redundancia a nivel de bit. Los códigos aportan distinta capacidad de corrección de errores según la redundancia que se añada. Algunos ejemplos de estos códigos son Hamming, Golay y BCH.

FEC permite la recuperación sin necesidad de retransmisiones, y su uso tiene sentido cuando las aplicaciones sean sensibles a los retardos que puedan provocar las retransmisiones. Este método tiene un buen comportamiento cuando las pérdidas son dispersas en el tiempo. En [11] se realiza una evaluación del comportamiento de FEC (tanto en fuentes FEC como en las que no lo son) con tres modelos de tráfico distinto, y se demuestra que, mientras para los dos modelos de tráfico homogéneo FEC no es efectivo, en el caso de las fuentes de vídeo (tráfico heterogéneo de vídeo o tráfico a ráfagas) FEC reduce las pérdidas en varios órdenes de magnitud. A cambio, debe incrementarse el tráfico en la red en una magnitud del orden del número de células redundantes generadas por el código FEC[3].

En cuanto a las necesidades de buffers de almacenamiento en los codificadores y decodificadores, hay que citar [11] que pueden ser las mismas, tanto con FEC como sin el. En el caso del buffer de los receptores depende del método que se emplee, ya que si se usa reensamblado de células en grandes unidades, deberá aportarse un buffer de reensamblado que sea suficientemente grande como para mantener ese bloque de células. En cambio, si el nodo receptor opera en modo *cut-through*, y cada célula contiene suficiente información de cabecera para determinar su posición, una célula de entrada podrá ser colocada en la aplicación del nodo receptor sin necesidad de buffers intermedios para la decodificación.

[3] Destacar que FEC es capaz de doblar su potencia como código corrector reemplazando errores por *erasures* que, como sabemos, tienen el error en una posición conocida.

RSE (código Reed-Solomon basado en código corrector de ráfagas de *erasures*) es un conocido sistema FEC que, partiendo de *d* células de datos como entrada, genera *r* células redundantes, dando como salida *d+r* células según muestra la *Figura 3.2.*

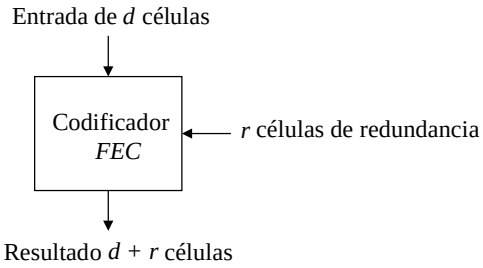

Figura 3.2. *Funcionamiento FEC*

El total de las *d+r* células son transmitidas al receptor que sólo necesitará decodificarlas si se reciben menos de las *d* células de datos originales. Sólo *d*, de las *d+r* células, son suficientes para recuperar las *d* células ATM iniciales. En ATM la unidad de detección de error y de recuperación de pérdidas es el bloque, que se define como un grupo de *d* células a partir de las cuáles se generan las *r* células redundantes. Un bloque de *d+r* células se pierde cuando se pierden más de *r* de las *d+r* células.

RSE es capaz de corregir *erasures* pero no errores. A cambio, su algoritmo es más sencillo y puede emplearse el mismo hardware en el codificador y en el decodificador. También trata las células de datos como texto plano sin ningún tipo de modificación, por lo que no es necesario esperar a que acabe todo el proceso de codificación para iniciar su transmisión, por lo que la decodificación es más rápida, al contrario que otros códigos que se encargan de cifrar los datos, aportando a éstos mayor seguridad. RSE puede ser implementado en un solo circuito y alcanzar throughputs entre 400 Mbps y 1 Gbps en función del tamaño de símbolo que se emplee. Usar símbolos de mayor tamaño (16 ó 32 bits en lugar de 8) supone alcanzar tiempos de codificación mucho menores, lo que afecta positivamente a la eficiencia de RSE.

Aunque existen distintas propuestas sobre el lugar de los árboles de distribución multicast en los que se debe aplicar la técnica FEC, lo que sí está demostrado es que este mecanismo decrementa la pérdida de paquetes en los árboles y, por tanto, también decrementa o amortigua el efecto negativo de la implosión en los emisores que es el mayor inconveniente para obtener la fiabilidad en las transmisiones multicast.

En grupos pequeños de usuarios se ha empleado FEC parcial, pero la eficacia de este método se aprecia a medida que se incrementa el número de receptores del grupo multicast, o cuando desciende el número de enlaces compartidos.

Diversos estudios realizados [16] demuestran que, cuando se aplica FEC a todos los enlaces de un árbol multicast, se observa lo siguiente:

- Para grupos pequeños la mejora de FEC incrementa cuando el número de receptores incrementa y el número de enlaces compartidos decrece.
- Para grupos grandes la mejora de FEC es mayor que para grupos pequeños y es independiente del número de receptores.

Cuando FEC se aplica a cualquiera de los enlaces del árbol multicast de una LAN o WAN se obtienen los siguientes resultados:

- Para grupos pequeños la mayor mejora de FEC se consigue cuando se aplica en la parte donde los enlaces tienen mayor probabilidad de pérdidas.
- Para grupos grandes esto sólo es cierto con un número bajo de receptores de la LAN. Si el número de receptores de la LAN es grande, FEC debería aplicarse a los enlaces de la LAN.

Minimizando el grado de solapamiento entre los paths de un árbol multicast se obtendrá un servicio multicast más fiable.

La referencia [10] demuestra cómo las comunicaciones multicast fiables son el área donde FEC puede ser más beneficioso que ARQ ya que éste escala mal cuando crece el número de receptores por los siguientes motivos:

- El emisor debe cargar con la respuesta de un gran número de ACK o NACK provenientes desde los receptores (fenómeno de la implosión de los ACK).
- Como el número de receptores crece, la probabilidad de pérdidas entre distintos receptores deja de estar relacionada entre sí, lo que causa que, si no todos, al menos una gran mayoría de los paquetes deba ser retransmitido con gran impacto en el rendimiento de las comunicaciones.

Aunque FEC suele implementarse mediante técnicas hardware usando codificadores y decodificadores, también es implementable mediante software sin pagar demasiado overhead. En el caso de IP, se emplean incluso técnicas de separación de canales apartando por uno de ellos las retransmisiones de datos con un servidor dedicado a esas retransmisiones y, por otro, la escucha de receptores que tienen pérdidas.

Hace ya algún tiempo que se intentaron aprovechar los beneficios de la recuperación de errores FEC en las redes ATM [11,12]. Sin embargo, parece que se ha prestado escasa atención a estas investigaciones. Las propuestas estándares de ITU-T [3,4] distinguen entre AAL-1 y AAL-5 como las capas adecuadas donde aplicar las técnicas FEC para la recuperación de errores. Tanto AAL-1 como AAL-5 emplean SSCS con FEC como control de errores para servicios en tiempo real. A continuación se va a describir brevemente la propuesta AAL-5.

3.5.1. FEC APLICADO A AAL-5

Las investigaciones para incluir técnicas FEC en AAL-5 [12] proponen una capa FEC-SSCS (FEC-Service Specific Convergence Sublayer) situada en la parte superior de AAL-5 y/o AAL-3/4, justo encima de la capa CPCS (Common Part Convergence Sublayer), tanto en emisores como receptores con la intención de obtener un rendimiento más elevado con menor latencia y mayor fiabilidad en la entrega de datos extremo-extremo.

Según muestra la *Figura 3.3*, cuando la entidad CPCS destino detecta un error (bit erróneo y/o célula perdida) al recibir una AAL-SDU, la entidad FEC-SSCS intenta recuperar el dato original enviado desde la CPCS fuente usando el algoritmo FEC.

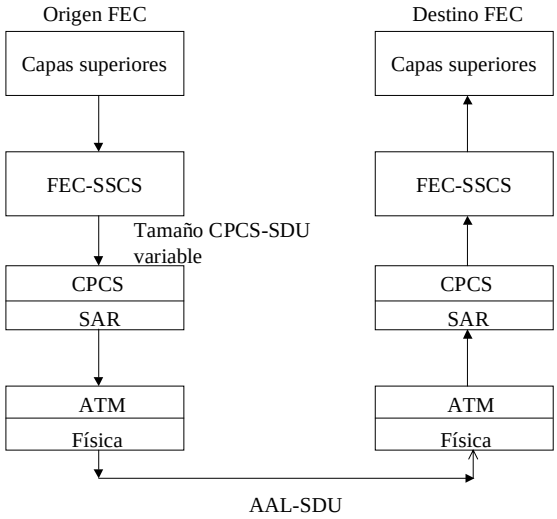

Figura 3.3. *Protocolo FEC-SSCS*

La propuesta distingue tres modos de operación relacionados con los tres tipos de errores distintos que es capaz de recuperar:

- SEC (Symbol Error Correction): Sólo recupera bits erróneos en los símbolos (unidades de datos definidas por el algoritmo FEC, típicamente de 8 ó 16 bits). La desaparición de símbolos o células no es tratada.
- SLC (Symbol Loss Correction): Sólo recupera símbolos que han desaparecido por pérdidas de células. No se recuperan bits erróneos.

- SEAL (Symbol Error And Loss correction): Recupera bits erróneos en los símbolos, y detecta la ausencia de símbolos porque la red haya tirado células.

Cada uno de estos modos de operación puede ser elegido por la aplicación de la capa superior que, como puede observarse en el modelo de referencia del protocolo, cuenta con las primitivas necesarias entre cada una de las capas. Los tres modos de operación ofrecen servicios como: indicación de pérdida de células; indicación de bit erróneo; negociación de los parámetros del algoritmo de FEC; ajuste de la longitud de los datos redundantes que se están enviando; tamaño variable de CPCS-SDU y uso de técnicas pipeline en las fuentes FEC-SSCS al transferir las FEC-SSCS-PDU.

3.5.2. FEC-SSCS EN MODO ATM NATIVO

Las propuestas estándares, tanto ITU-T como ATM-Forum, distinguen entre diferentes métodos de corrección de errores, sin embargo, ninguna de ellas procesa directamente células ATM nativas de 53 octetos, incurriendo en la ineficiencia que supone el tener que gestionar PDUs y arrastrar, en algunos casos, importantes overheads que pueden hacer del método FEC ineficiente en algunos casos.

Por ejemplo, en [9], el método de corrección de errores en los bits para AAL-1 emplea en la CS emisora códigos de Reed-Solomon de 4 octetos que se añaden a los 124 octetos de datos que entran provenientes de la capa superior. El método es capaz de corregir dos octetos con errores en cada trama FEC de 128 octetos, e incurre en un overhead del 3,1 % y un retardo en el receptor de aproximadamente 3 células. En la CS emisora cada PDU es un bloque FEC formado por 47 tramas FEC de 128 octetos cada una.

En el método de corrección de errores en bits y pérdidas de células, los bloques de 128 (124 de datos + 4 de FEC) octetos resultantes se envían al entrelazador de octetos para generar la matriz de entrelazado que está formada por 47 líneas de 128 columnas cada una. Cada línea equivale a una SAR-PDU y la matriz completa es una CS-PDU.

El tercer método de corrección tampoco gestiona células ATM nativas, lo que acaba generando PDU que es necesario adaptar en las capas AAL-5 y acaban provocando el overhead que podría evitarse trabajando en modo ATM nativo.

Del mismo modo, la propuesta del draft del ATM-Forum [8] para AAL-5 tampoco se centra en la obtención de PDU cercanas a los tamaños de células ATM nativas que eviten el desaprovechamiento del throughput.

3.6. TÉCNICAS HÍBRIDAS FEC Y ARQ

Tradicionalmente, la mayoría de investigaciones en materia de protocolos multicast fiables se han centrado en la recuperación de errores mediante técnicas ARQ. Sin embargo, el inconveniente de este método es su escasa escalabilidad cuando el número de componentes del grupo multicast empieza a crecer. Para solventar este inconveniente se han propuesto en la literatura diversas técnicas para aportar a ARQ la escalabilidad que le falta, entre ellas las dos siguientes [16]: establecimiento de asincronía entre receptores con la intención de evitar la implosión y, por otro lado, la aplicación de jerarquías a los árboles de distribución multicast. No obstante, la falta de escalabilidad hace que ARQ sea usado en protocolos unicast y se ha propuesto FEC para incrementar la fiabilidad en las comunicaciones multipunto.

Partiendo de la premisa de la imposibilidad de conseguir la fiabilidad total, hay que destacar que, en comunicaciones punto-a-punto, se ha logrado haciendo que el receptor envíe mensajes ACK positivos al emisor. Para los paquetes de datos recibidos se emplea confirmación positiva y confirmación negativa para los paquetes perdidos. En ambas situaciones sigue existiendo la implosión que, en el caso del multicast fiable, se evita haciendo que los receptores envíen NACK al emisor cuando se han perdido algunos datos.

ARQ es un mecanismo de control de errores basado en retransmisiones, de utilidad en situaciones donde existan grupos pequeños, en aplicaciones no interactivas, cuando predominan las pérdidas en enlaces compartidos de los árboles multicast, o cuando la probabilidad de pérdidas no es homogénea. FEC, en cambio, es interesante cuando existen grupos grandes, predominan las pérdidas individuales, la probabilidad de pérdidas es homogénea o los buffers son limitados.

Las referencias [7,17-20] aportan importantes ideas y justifican la aplicación de FEC como la técnica apropiada para las transferencias multicast fiables en redes IP. No obstante, muchas de las ideas pueden ser aprovechables para las redes de tecnología ATM.

La aplicación de técnicas correctoras de errores al tráfico multipunto a través de métodos híbridos FEC-ARQ es, sin lugar a dudas, una de las partes más complejas por todas las implicaciones del multipunto. La idea general va en la línea de aplicar FEC entre emisor y receptores pero teniendo en cuenta que en determinados puntos de la red puede tener sentido aplicar ARQ, por ejemplo, cuando no quede otro remedio que retransmitir en los extremos. Hemos analizado este problema y consideramos que si es así se evitaría la implosión hacia arriba en el árbol formando subgrupos para que, cuando un conmutador vecino al que ha detectado el error haya sido capaz de reconstruir un error, no sea necesario retransmitirlo desde el origen. En esta situación hay que considerar los problemas de la heterogeneidad de receptores y de los distintos tramos de red.

En general, podemos considerar el escenario que se muestra en la *Figura 3.4*, donde se indica que la mayor parte de pérdidas de células se producen en las troncales de las redes y no en los nodos extremos de la comunicación. Para amortiguar las pérdidas se puede usar FEC, pero cuando se detecten pérdidas en los receptores emplearíamos NACK para avisar al conmutador correspondiente, que se encargará de las retransmisiones lo más locales posibles. Para evitar la implosión en el emisor, propondríamos la existencia de buffers en los conmutadores que permitan que entre conmutadores vecinos puedan realizarse las retransmisiones (mediante NACK) sin necesidad de afectar al emisor hasta que no sea necesario. Hemos de destacar que, una vez reflexionados estos planteamientos y analizados más en detalle, nos ha conducido a algunos de los fundamentos de la arquitectura TAP que van a ser descritos en capítulos siguientes.

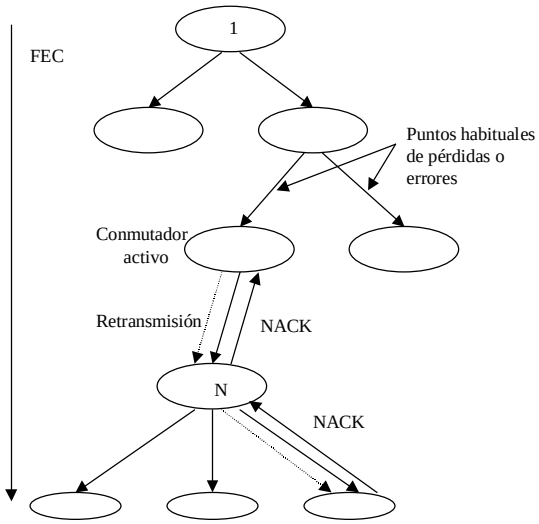

Figura 3.4. *FEC-NACK híbrido en multicast*

3.7. TAP Y GOS

En secciones anteriores hemos destacado la diferencia entre fiabilidad total y lo que nosotros hemos denominado GoS. Así hemos querido aclarar que nuestro objetivo no es el de aportar fiabilidad a las células o PDUs ATM desde el punto de vista de las corrupciones de datos que puedan experimentar, sino que nuestro trabajo pretende solventar los problemas relativos a las congestiones de los conmutadores ya que éstas son la causa más habitual de errores.

A la vista de lo comentado en este capítulo podemos decir que nuestra propuesta de GoS tiene escasa relación con las técnicas que recurren a códigos redundantes, ya sea CRC o FEC, ya que éstas están pensadas para aportar fiabilidad con respecto a la corrupción de datos pero, por sí solas, no son capaces de solventar los problemas provocados por las congestiones. Sin embargo ACK+ y NACK, a pesar de ser dos técnicas menos eficientes de cara a la fiabilidad, tienen la ventaja de poder ser usadas para el control de congestión ya que, mediante temporizadores y números de secuencia, podemos conocer cuándo un nodo de la red está perdiendo células. Esta es precisamente la característica que hemos aprovechado para implementar la GoS a las conexiones que el administrador de la red decida configurar como privilegiadas.

La eficiencia de NACK sobre ACK+ y toda la serie de ventajas que hemos comentado en puntos anteriores nos ha servido para elegir los acuses de recibo negativos como método de notificación de pérdidas de células desde los receptores a los emisores. Para poder aplicar el método NACK únicamente necesitamos soportar un mecanismo que nos permita introducir los números de secuencia que las células no soportan de

forma estándar. Para solventar este problema proponemos una reinterpretación de los campos de las PDU de AAL-5 que hemos comentado anteriormente. Lo que hacemos es emplear los campos CPCS-UU y CPI, cada uno de 1 byte, para disponer de 16 bits que nos permiten disponer de una ventana de 65.535 PDU numeradas en secuencia para cada conexión. La *Figura 3.5* presenta la posición en que aparecen los identificadores de PDU (PDUid) dentro de las colas de las PDU de AAL-5.

Como es conocido, AAL-5 fue desarrollada para el soporte de transferencias no garantizadas de tramas de datos de usuario donde la pérdida y corrupción de Common Part Convergence Sublayer Service Data Unit (CPCS-SDU) no puede solventarse con retransmisiones [4]. En TAP proponemos Extended AAL type 5 (EAAL-5) como extensión y mejora de las características de AAL-5 nativo. EAAL-5 es parte de TAP y soporta GoS con retransmisiones y es también compatible con AAL-5 nativo. Como veremos en capítulos posteriores proponemos también un mecanismo para aprovechar los periodos de inactividad (*idle*) de las fuentes para retransmitir las CPCS-PDU de EAAL-5.

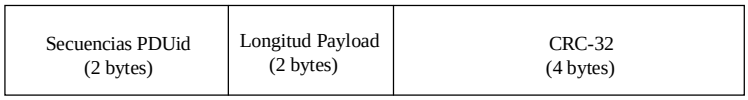

Figura 3.5. *Cola de CPCS-PDU-AAL-5 con PDUid de AAL-5*

Destacamos que los PDUid se mantienen extremo-extremo para cada una de las conexiones sin ningún tipo de renumeración en los conmutadores intermedios de la red para evitar recálculos de CRC en esos nodos intermedios. Cuando el emisor llega al valor final de la secuencia de PDU comienza nuevamente por el principio.

Cuando algún conmutador entra en situación de congestión empleará el valor de PDUid (además de otros valores) para solicitar la retransmisión de las PDU que han sufrido las congestiones. Veremos más adelante, cuando describamos el protocolo completo, que empleamos células RM del tipo ABR para solicitar las retransmisiones y también estudiaremos la forma en que evitamos la implosión de los emisores haciendo que las retransmisiones las atiendan los conmutadores intermedios que soportan la arquitectura TAP.

Con el mecanismo que acabamos de describir no podemos garantizar fiabilidad ni GoS completa, pero demostraremos que gran parte de las situaciones de congestión pueden ser resueltas con retransmisiones locales NACK, cosa que ni CRC ni FEC pueden garantizar.

3.8. CONCLUSIONES

Las células ATM aportan un mecanismo de control de errores basado en el campo HEC. Sin embargo, los errores producidos en las transferencias a través de la red no son el principal problema que pueden experimentar las células. Un problema más grave, y menos predecible, es el de la congestión provocada en los conmutadores cuando se les exige un rendimiento superior al que son capaces de ofrecer. Es decir, los conmutadores pueden experimentar congestiones cuando las fuentes de tráfico no cumplen sus contratos de tráfico y producen, individualmente o de forma conjunta, más células por segundo que las que alguno de los conmutadores intermedios es capaz de procesar.

Hemos comprobado que, mientras ARQ añade latencia (debida al coste de los ACK+ y NACK) e *implosión* (congestión de los emisores para atender excesivas retransmisiones), FEC añade sobrecarga por cabeceras y también código redundante cuando la red está experimentando congestiones. Por tanto, ARQ no es apropiado para aplicaciones que requieren baja latencia, y FEC se comporta peor en redes con bajo ancho de banda o que experimentan congestiones habituales. En nuestra arquitectura adoptamos ARQ con NACK usando células Resource Management (RM) para aliviar el efecto de la implosión.

TAP ofrece Garantía de Servicio cuando la red está perdiendo células ATM y para ello aprovecha los periodos de inactividad en las fuentes de tráfico, momentos en los cuales se realizan las retransmisiones de las CPCS-PDU-EAAL-5.

En suma, en la arquitectura TAP la fiabilidad es ofrecida por el campo Header Error Control (HEC) de 8 bits de la cabecera de las células ATM y por el Cyclic Redundancy Code (CRC) de la subcapa Common Sublayer-Protocol Data Unit (CS-PDU). El control de errores se realiza extremo-a-extremo por los equipos terminales de la comunicación. Sin embargo la GoS es conseguida modificando las colas de PDU-AAL-5 y mediante NACK con células estándares RM. Las congestiones son resueltas en los conmutadores adyacentes o vecinos al conmutador congestionado.

REFERENCIAS

[1] R. Steinmetz, and L. C. Wolf, "Quality of Service: Where are We ?," *Fifth International Workshop on Quality of Service IWQOS'97*, pp.211-221, (1997).

[2] ____ Rec. I.361, "B-ISDN ATM Layer Specification," *ITU-T*, 11/1995.

[3] ____ Rec. I.363.1, "B-ISDN ATM Adaptation Layer Specification, Type 1," *ITU-T*, 08/1996.

[4] ____ Rec. I.363.5, "B-ISDN ATM Adaptation Layer Specification, Type 5," *ITU-T*, 08/1996.

[5] ____ Rec. I.371.1, "Traffic Control and Congestion Control in B-ISDN," *ITU-T*, 06/1997.

[6] M. de Prycker, "Asynchronous Transfer Mode. Solution for Broadband ISDN (3ª Ed.)," *Ed. Prentice Hall.*,(1995).

[7] Stephan Block, Ken Chen, Philippe Godlewski, and Ahmed Serhrouchni, "Design and Implementation of a Transport Layer Protocol for Reliable Multicast Communication," *Université Paris, http://www.enst.fr/~block/srmtp,* (1998).

[8] Jörg Nonnenmacher, M. Lacher, M. Jung, E. Biersack, and G. Carle, "How bad is Reliable Multicast without Local Recovery?," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE,* Vol. 3 pp. 972-979, (1998).

[9] Dan Rubenstein, Sneha Kasera, Don Towsley, and Jim Kurose, "Improving Reliable Multicast Using Active Parity Encoding Services (APES*)", Technical Report 98-79, Department of Computer Science, University of Massachusetts*, July, (1998).

[10] L. Rizzo, "On the feasibility of software FEC," *http://www.iet.unipi.it/~luigi*, Universita di Pisa, (1997).

[11] Ernst W. Biersack, "Performance Evaluation of Forward Error Correction in an ATM Environment," *IEEE Journal on Selected Areas in Comm*., vol. 11, No. 4, pp. 631-640, May. (1993).

[12] H. Esaki, G. Carle, T. Dwight, A. Guha, K. Tsunoda, and K. Kanai, "Proposal for Specification of FEC-SSCS for AAL Type 5*," Contribution ATMF/95-0326 R2, ATM Forum Technical Committee*, (October 1995).

[13] Marsh, S.P., "Formalising Trust as a Computacional Concept, in Computing Science and Mathematics," *University of Stirling, PhD Thesis http://ai.iit.nrc.ca/~steve/pubs/Trust.ps.Z,* (1994).

[14] Wilhelm, U.G., S. Staamann, and L. Buttyan, "On the problem of trust in mobile agent systems," *IEEE Symposium on Network And Distributed System Security*, 1999. *http://icawww.epfl.ch/buttyan/publications/NDSS98.ps* (1999).

[15] Kini, A. and J. Choobineh, "Trust in Electronic Commerce: Definition and Theoretical Considerations", *31st Annual Hawaii International Conference on System Sciences*. 1998. Hawaii. *http://ieeexplore.ieee.org/ie14/5217/14270/00655251.pdf* (1998).

[16] Jörg Nonnenmacher and Ernst Biersack "Reliable Multicast: Where to use FEC," *Proceedings of IFIP 5th International Workshop*, (1996).

[17] Georg Carle and Ernst W. Biersack, "Survey of Error Recovery Techniques for IP-based Audio-Visual Multicast Applications," *IEEE Networks*, (1995).

[18] Roger G. Kermode "Scoped Hybrid Automatic Repeat ReQuest with Forward Error Correction (SHARQFEC)," *ACM SIGCOMM'98*, (1998).

[19] Dan Rubenstein, Jim Kurose, and Don Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction," *Technical Report 98-19.* Department of Computer Science University of Massachusetts (1998).

[20] Jörg Nonnenmacher, Ernst Biersack and Don Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission" *IEEE Transaction on Networking*, (1998).

[21] Don Towsley, Jim Kurose, and Sridhar Pingali, "A comparison of sender-Initiated Receiver-Initiated Reliable Multicast Protocols," *IEEE Journal os Selected Areas in Communications*, (April 1997).

CAPÍTULO 4

CONTROL DE CONGESTIÓN Y GESTIÓN JUSTA DE COLAS

4.1. INTRODUCCIÓN

Otro importante aspecto relacionado con el control de congestión al que nos enfrentamos en esta tesis es el que surge de las diferentes características de tráfico que tienen las fuentes. La caracterización del tráfico, como sabemos, es una de las ventajas de ATM, que permite la integración de todo tipo de CoS. Sin embargo, esto puede ser también una importante fuente de problemas por la posibilidad que existan flujos de entrada a los conmutadores que generen situaciones de injusticia con respecto a los recursos de otras fuentes. Esa posibilidad de mal comportamiento de unas fuentes exigentes con respecto a otras que lo son menos puede acabar generando situaciones de inanición y también de congestión que hay que intentar solventar.

En muchas tecnologías de redes el control de congestión se consigue confiando en protocolos extremo-a-extremo como TCP. Esto permite disponer de routers y conmutadores muy sencillos, sin embargo, pueden aparecer situaciones críticas si todos los extremos de la comunicación no cooperan, ya que -de otro modo- los flujos con buen comportamiento no quedarán bien protegidos de los flujos más ambiciosos que, en el límite, pueden acabar quedándose con todos los recursos de la red. Además, los extremos de la comunicación deberán implementar algoritmos de control homogéneos, para evitar que aquellos usuarios que empleen algoritmos de control más estrictos puedan autogenerar situaciones de injusticia respecto a aquellas fuentes de tráfico con control menos riguroso.

En nuestra propuesta de arquitectura TAP se ve clara la necesidad de aportar mecanismos de justicia ya que estamos proponiendo fuentes privilegiadas de tráfico a las que se les ofrece GoS, lo que puede acabar degenerando en situaciones de injusticia para aquellas fuentes que no se definan como privilegiadas. Así, una alternativa al control de congestión en los extremos es diseñar conmutadores que soporten asignaciones de ancho de banda justas para proteger a las fuentes no privilegiadas de las que sí lo son. De este modo, cada fuente tendrá garantizados sus recursos. Pero uno de los principales problemas al que nos enfrentamos es que los algoritmos propuestos para aportar la necesaria justicia son demasiado complejos ya que deben realizar la gestión separada (*per-flow*) de flujos. Siendo más precisos, un router o un conmutador que implemente estos algoritmos deberá realizar: una clasificación separada de paquetes, una gestión separada de buffers para cada flujo y, en la mayoría de los casos, también una planificación separada de flujos. Toda esta complejidad añadida puede hacer peligrar otros parámetros de QoS como el *delay* y *jitter* que son un importante requerimiento para fuentes con necesidades de alta velocidad como las aplicaciones de tiempo real.

Como se ha comentado en capítulos previos, una de las principales características de ATM es su capacidad para garantizar la QoS a diversos tipos de aplicaciones, tanto de tiempo real como no tiempo real. Para las aplicaciones de tiempo real deben garantizarse, entre otros parámetros de tráficos, un *delay* y un *jitter* máximos negociados en el establecimiento de la conexión. Sin embargo, las aplicaciones pensadas para la transmisión de datos no tienen excesivos requerimientos de *delay* pero sí en lo referente a la justicia de unas fuentes de tráfico con respecto a otras que puedan consumir excesivos recursos.

Existen diferentes planteamientos en cuanto a la arquitectura de los conmutadores ATM, pero todos ellos cuentan con la inclusión de colas que, en esencia, se usan para aplicar diferentes esquemas de gestión de

tráfico. Precisamente este planteamiento explica la importancia de las políticas de gestión de colas.

En general, los conmutadores ATM enrutan las células desde un conjunto de puertos de entrada hasta un conjunto de puertos de salida basados en los identificadores de flujo VPI/VCI. En el puerto de salida de un conmutador ATM un controlador de puerto multiplexa células desde diferentes flujos de entrada en la línea de salida. Los puertos de salida de los conmutadores ATM son gobernados por una política de planificación como FIFO, que decide qué células desencolar de la cola de salida. Las políticas de planificación pueden ser clasificadas en *work-conserving* o *non work-conserving*. Una política es *work conserving* si nunca deja el enlace de salida en estado de inactividad mientras haya células dentro de la cola de salida. Al contrario, una política *non work-conserving* puede dejar el enlace de salida en estado de inactividad incluso si la cola de salida no está vacía. La *Tabla 4.1* presenta la clasificación de algunas de las más conocidas políticas de planificación.

TABLA 4.1
MECANISMOS DE CONTROL BASADOS EN VELOCIDAD

Servicios work-conserving	**Servicios non-work-conserving**
Weighted Fair Queueing	Jitter Earliest-Due-Date
Virtual Clock	Stop-and-Go
Delay Earliest-Due-Date	Hierarchical Round-Robin
Self-Clocked Fair Queueing	

Todas las políticas *work-conserving* tienen, para un conjunto dado de patrón de llegada de células, idéntico retardo medio de células y necesidades máximas de buffer. Una política *non work-conserving* puede necesitar mayor retardo medio de células y necesidades de buffer máximo y medio que una política *work-conserving*. Por otro lado, las políticas *work-conserving* tienden a incrementar las ráfagas de tráfico, mientras las políticas *non work-conserving* pueden ser usadas para limitar el tráfico a ráfagas. En líneas generales las políticas *work-conserving* son más fáciles de implementar y requieren una gestión de buffers más sencilla.

Los sistemas *work-conserving* envían los paquetes una vez que el servidor ha completado el servicio. Por tanto, el servidor nunca queda en estado de inactividad si existen trabajos en una cola del sistema. Los dos ejemplos más claros y clásicos de los sistemas *work-conserving* son FIFO y LIFO. Los esquemas *non work conserving* se caracterizan porque las servidores esperan un espacio de tiempo aleatorio antes de servir el siguiente paquete de una cola, incluso si hay paquetes esperando en las colas.

En ATM las aplicaciones diferentes tienen diferentes requerimientos de tiempo y ancho de banda. Para asegurar que cada aplicación tenga la QoS requerida, la red debe realizar alguna forma de control de admisión a los flujos que intentan acceder a los conmutadores. El control de admisión deberá asegurar que existen suficientes recursos en la red antes de conceder al flujo la QoS requerida, para no comprometer la QoS de las conexiones que ya están dentro de la red. Para garantizar que cada conexión puede tener la QoS solicitada se aplica la función CAC (Call Admission Control). Existen muchas formas de implementar la CAC, sin embargo, muchas de ellas sólo ofrecen garantías estadísticas, es decir, en términos de retardo medio de célula. Para tráfico de tiempo real, donde la QoS debe tener una garantía absoluta, este tipo de CAC no son aplicables. En lugar de ello, una garantía determinista garantiza que nunca se pierde el plazo de entrega.

En general, suele distinguirse entre dos tipos de planteamientos en cuanto a la llegada de los datos: garantía estadística y determinística. La garantía estadística promete que un porcentaje de los datos llegará con un retardo *D*, o que una media del ancho de banda está disponible para el flujo. Sin embargo, la garantía determinística toma la forma de promesa de que todos los datos llegarán con un retardo *D* respecto a la transmisión. También garantiza que un flujo accederá, como poco, a *X* bits por segundo o que las pérdidas no excederán de un cierto valor.

Las colas pueden ser gestionadas de muy diversas formas y el método elegido para hacer esta gestión tiene muy diferentes efectos en el tráfico que fluye a través de las colas. Otro importante aspecto relacionado con las colas es que cada esquema va a suponer en cada conmutador una serie de requerimientos para realizar la implementación del sistema de colas.

Otro aspecto directamente relacionado con este tema es la integración con diferentes redes. Esto causa que, aunque los conmutadores y/o routers gestionen su tráfico perfectamente, el tráfico puede no tener garantizado el buen rendimiento.

Para acabar de situar este importante aspecto de gestión de tráfico destacamos su relación con el control de velocidad. Así, puede realizarse la siguiente división:

- Las redes de datos de conmutación de paquetes suelen usar mecanismos FIFO y control de flujo basado en ventana.
- Por otro lado, las redes de comunicación multimedia suelen usar mecanismos basados en velocidad. Este planteamiento puede clasificarse según lo que puede observarse en la *Tabla 4.2*

Las técnicas de encolamiento *per-VC*[1], que aplican diferentes pesos a las colas, se han demostrado como un mecanismo apropiado para satisfacer los requerimientos de *delay*, *jitter* y justicia comentados al inicio, y se han propuesto múltiples algoritmos de encolamientos *per-VC* basados en velocidades. Algunos de los ejemplos más significativos de estos mecanismos son WFQ (Weighted Fair Queueing) [2], VirtualClock [20], SCFQ (Self-Clocked Fair Queueing) [6], Delay-EDD (Delay Earliest Due Date), Jitter-EDD [21] y HOL-EDD (Head-Of-the-Line EDD) [22]. Todos estos algoritmos, salvo Jitter-EDD, son *work-conserving*, lo que implica que el servidor no descansa mientras existan paquetes en espera de ser procesados.

La *Tabla 4.2* presenta una división de las políticas de planificación *work-conserving* más interesantes, que pueden ser clasificadas en políticas simples en cuanto a su implementación y eficiencia, y en políticas basadas en velocidad, en las cuales a cada flujo se asigna una velocidad de servicio o un peso que determina cuántos recursos se deben asignar al flujo. Como regla general, las políticas basadas en pesos aproximan la disciplina de servicio ideal GPS (Generalized Processor Sharing) y, para todas ellas, es común que el tiempo de respuesta sea directamente dependiente de la velocidad de servicio asignada a cada flujo y la velocidad de servicio se corresponde con el ancho de banda asignado. Debe destacarse que la implementación de las técnicas basadas en velocidad son mucho más complejas de implementar.

TABLA 4.2
CLASIFICACIÓN DE POLÍTICAS DE PLANIFICACIÓN WORK-CONSERVING

Políticas de planificación sencilla	Políticas de planificación basadas en velocidad
FIFO	GPS
Priority Queueing	Virtual Clock (VC)
Earliest Deadline First (EDF)	Weighted Round-Robin (WRR)
Response-Time Analysis (RTA)	WFQ
Calculus for Network Delays (CND)	Multirate PGPS

La *Figura 4.1* muestra los diferentes comportamientos y evolución de los mecanismos *fair queueing* que comentamos brevemente:

- La propuesta (1) es la conocida como solución original de Nagle [1] que se caracteriza por el aislamiento de las fuentes de tráfico que se comportan incorrectamente. Esta solución presenta dos limitaciones destacables: en primer lugar, se ignora la longitud de los paquetes que llegan a las colas que, como podemos ver, pueden ser de diferente tamaño teniendo todos un mismo tiempo de ciclo y, en segundo lugar, este esquema es sensible al patrón de llegadas de paquetes.
- La solución (2) propuesta por Demers, Keshav y Shenker [2] parte de paquetes todos de igual tamaño (1 bit), de forma que se aplica justicia realizando esperas de *n-1* bits antes de que una fuente vuelva a enviar un nuevo paquete, lo que presenta la limitación de que cada fuente tiene la misma fracción de ancho de banda, sin permitir la caracterización del tráfico.

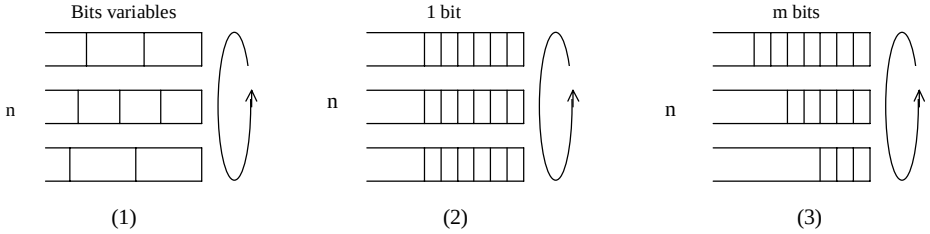

Figura 4.1. *Esquemas justos de colas*

[1] Las técnicas de colas per-VC se caracterizan por emplear colas separadas para cada VC. En nuestro caso tendremos colas separadas, tanto para transmisiones garantizadas, como para las que no lo son.

- El esquema (3) coincide con el conocido WFQ (Weighted Fair Queueing) caracterizado porque las fuentes pueden enviar diferente número de paquetes en cada ciclo. En este caso *m* es mayor que *n* y se establecen ciclos de *m* bits cada uno. Parekh [3,4] en su tesis de 1992 realiza una demostración de las garantías de funcionamiento de esta propuesta.

A continuación vamos a realizar una breve descripción de algunas de las técnicas de control de congestión basadas en la justicia de colas, describiendo sus ventajas e inconvenientes para concluir con la presentación general de nuestra aportación que será explicada en detalle en capítulos siguientes.

4.2. FAIR QUEUEING

Una de las disciplinas de servicio de paquetes a los puntos de encolado de los conmutadores es la PS (Processor Sharing) que emplea colas FIFO separadas para cada sesión que comparte un enlace. En PS todas las sesiones en espera reciben igual tamaño de ancho de banda, independientemente del tamaño de los paquetes que tengan. PS es una disciplina ideal cuando el servidor es capaz de servir *N* sesiones simultáneas. En realidad, el servidor envía un paquete cada vez. Precisamente, [2] propone un algoritmo de aproximación a paquetes de PS llamado FQ (Fair Queueing) con asignación justa del ancho de banda y protección contra las fuentes malintencionadas. Además, para los servidores que se ajustan o aproximan la disciplina de servicio PS, las fuentes pueden medir más ajustadamente el estado de la red. De este modo queremos destacar que PS es una disciplina de servicio que permite el diseño y la implementación de algoritmos de congestión muy robustos.

Los algoritmos FQ (Fair Queueing) [2,3,4] aportan atractivas ventajas y han sido diseñados para que routers y conmutadores ofrezcan el control de congestión y para la asignación justa de ancho de banda. Sin embargo, su coste de implementación completa en una red puede ser no aceptable para aplicaciones de alta velocidad ya que, aparte de la complejidad de programación, requieren que el mecanismo FQ mantenga información de estado, gestione los buffers de entrada, realice planificación de los paquetes y, todo ello, sobre la base de la separación de los flujos.

No obstante, varias investigaciones han demostrado que las propuestas de control de congestión extremo-a-extremo pueden ser mejoradas de una forma sustancial si en los routers se aplican técnicas justas de asignación del ancho de banda disponible en la red ya que, además de proteger unas fuentes de otras, permiten que en la red puedan coexistir diversas políticas de control de congestión. Del mismo modo, varias de estas investigaciones también reconocen que la asignación justa de ancho de banda juega un papel necesario, aunque a veces no beneficioso, en el control de congestión. Así, la mayor parte de propuestas en este campo parten de dos premisas importantes, por un lado que los mecanismos justos de asignación de ancho de banda son necesarios, y por otro, que la complejidad de estos mecanismos es un importante factor para su adopción en las redes.

Lo que parece claro es que cualquier propuesta que realicemos en este sentido será bastante más compleja de implementar que el clásico encolamiento FIFO que ha sido el más usado tradicionalmente. Como es sabido, en FIFO los paquetes son servidos en el orden de llegada, y los buffers son gestionados siguiendo una sencilla estrategia *drop-tail* según la cual los paquetes entrantes son tirados cuando el buffer está lleno. Pero FIFO carece de la justicia que se buscó con FQ [2,3,4] y sus variantes [5,6,7] y con mecanismos de abandono por flujos como FRED (Flow Random Early Drop) [8]. En cualquiera de estas técnicas se requiere mantener el estado de los diversos flujos por separado, de forma que, para cada paquete que llega, hay que clasificarlo en su correspondiente flujo, actualizar las variables de estado de cada flujo y realizar operaciones (*encolar_paquete*, *tirar_paquete*, *priorizar_flujo*, etc) en función del estado de cada flujo.

RED (Random Early Detection) [9] sirve también los paquetes en el orden de llegada, pero la gestión del buffer es mucho más sofisticada que el *drop-tail* de FIFO. RED tira probabilísticamente paquetes largos antes de que el buffer se llene, ofreciendo indicación de congestión rápida a los flujos que pueden ser afectados antes de que se produzca el rebosamiento del buffer. De este modo RED mantiene dos umbrales en los buffers. Cuando la ocupación media del buffer es menor que el primer umbral, no se tira ningún paquete, pero cuando se supera el segundo umbral, los paquetes comienzan a ser tirados por la red. Cuando la ocupación del buffer se encuentra entre los dos umbrales, es cuando la probabilidad de tirar paquetes incrementa linealmente con la ocupación del buffer.

FRED (Flow Random Early Drop) [8] lo que hace es ampliar RED para ofrecer un cierto grado de asignación justa de ancho de banda. Para conseguir la justicia, FRED mantiene el estado de todos los flujos que tienen, como poco, un paquete en el buffer. Al contrario que en RED, donde la decisión de tirar paquetes se basa sólo en el estado del buffer, en FRED la decisión se basa también en el estado de los flujos. De este modo, FRED tira preferentemente los paquetes de un flujo al cual se le han tirado pocos paquetes

anteriormente, y que poseen una cola mayor que el tamaño medio de las colas. Este algoritmo tiene dos variantes, que se diferencian en que una de las versiones garantiza a cada flujo un número mínimo de buffers.

Todos los algoritmos comentados presentan diferentes niveles de complejidad. FRED se encarga de clasificar los flujos entrantes, mientras FIFO y RED no lo hacen. En suma, ninguno de ellos debe implementar ningún algoritmo de planificación de paquetes, sin embargo, todos aplican una sencilla planificación FIFO.

4.3. GPS, PGPS (WFQ) Y PFQ

Este apartado comenta brevemente el modelo de tráfico continuo ampliamente aceptado, para pasar luego a las propuestas de aproximación a este modelo para las redes de paquetes. De este modo tratamos de aclarar algunas confusiones en torno a estas disciplinas e identificamos algunos aspectos claves que luego empleamos en nuestra propia propuesta.

La disciplina GPS (Generalized Processor Sharing) tiene dos propiedades importantes: 1) puede garantizar un servicio de retardo limitado extremo-a-extremo a una sesión cuyo tráfico está controlado por leaky bucket; y 2) puede asegurar asignación justa de ancho de banda a todas las sesiones que tienen trabajos en espera, estén o no sometidas al control de tráfico. Mientras la primera propiedad es importante para soportar tráfico best-effort y servicios jerárquicos de enlace compartido, la segunda propiedad es la base para aportar GoS al tráfico.

Debido a que GPS usa un modelo fluido ideal que no puede ser realizado en el mundo real, se han propuesto varios algoritmos de paquetes como aproximación a GPS. En la literatura se denominan PGPS (Packet Generalized Processor Sharing) [3,4] a estas aproximaciones, aunque en otros casos [5,10] también se denominan como PFQ (Packet Fair Queueing). Entre los algoritmos que implementan GPS en redes de paquetes destaca WFQ (Weighted Fair Queueing) [2] que es actualmente el más popular método para ofrecer garantía de funcionamiento y es también conocido como PGPS (Packet Generalized Processor Sharing) [3,4].

Contrariamente a la idea que se ha extendido de forma generalizada, existen importantes diferencias entre los servicios ofrecidos por el sistema de paquetes WFQ y el sistema fluido GPS y estas diferencias son estudiadas en detalle en [5].

Precisamente, uno de los aspectos más importantes en el diseño de redes de servicios integrados es la elección de la disciplina de servicio de los paquetes en los puntos de encolado de la red. GPS es una de esas disciplinas, a la que se ha prestado una especial atención. GPS es una forma general de procesador *head-of-line* compartiendo disciplinas de servicio PS. Con PS tenemos una cola FIFO separada para cada sesión que comparte el mismo enlace. Durante un intervalo de tiempo, cuando hay exactamente N colas no vacías cada una de ellas asociada a un servicio compartido, el servidor GPS sirve los N paquetes de la cabecera de las colas simultáneamente, cada uno a una velocidad de N veces la velocidad del enlace. Mientras un servidor PS sirve todas las colas no vacías a la misma velocidad, GPS permite a diferentes sesiones tener diferentes servicios compartidos y sirve las colas no vacías en proporción al servicio compartido de las correspondientes sesiones.

Un servidor GPS, sirviendo N sesiones, se caracteriza por N números reales positivos $\phi_1, \phi_2,..., \phi_N$. Estos valores indican el tamaño relativo de servicio para cada sesión. El servidor trabaja a una velocidad fija r y es *work-conserving*. Sea $W_i(t_1, t_2)$ el tamaño de sesión i servido en el intervalo $[t_1, t_2]$, así, un servidor GPS se define como aquel para el cual

$$\frac{W_i(t_1,t_2)}{W_j(t_1,t_2)} \geq \frac{\phi_i}{\phi_j} \qquad j = 1,2,...,N \tag{1}$$

se mantiene para una sesión i que está pendiente durante el intervalo $[t_1, t_2]$. Si $B_{GPS}(\tau)$, el conjunto de sesiones pendientes en el tiempo τ, permanece inalterable durante el intervalo de tiempo $[t_1, t_2]$, la velocidad de servicio de la sesión i durante el intervalo será exactamente

$$r_i^*(t_1,t_2) = \frac{\phi_i}{\sum_{j \in B_{GPS}(t_1)} \phi_j} r \tag{2}$$

donde *r* es la velocidad del enlace. Como $B_{GPS}(t_1)$ es un subconjunto de todas las sesiones del servidor, puede verse como

$$r_i^*(t_1,t_2) \geq r_i \tag{3}$$

se mantiene, donde

$$r_i = \frac{\phi_i}{\sum_{j=1}^{N}\phi_j} r \tag{4}$$

Así, la sesión *i* tiene garantizada una mínima velocidad de servicio r_i durante un intervalo en el que está pendiente. Dada la longitud del intervalo de tiempo hasta llegar a cero, obtenemos la velocidad de servicio instantánea de la sesión $r_i^*(\tau)$.

Hay que destacar que GPS es un servidor ideal que no transmite paquetes como entidades. Esto asume que el servidor puede servir todas las sesiones pendientes simultáneamente y que el tráfico es infinitamente divisible. En un sistema de paquetes más realista sólo una sesión puede recibir servicio en un instante, y un paquete entero debe ser servido antes de servir otro. La *Figura 4.2* presenta un diagrama de tiempos de la llegada y transferencia de paquetes pertenecientes a la sesión *i* en un sistema GPS.

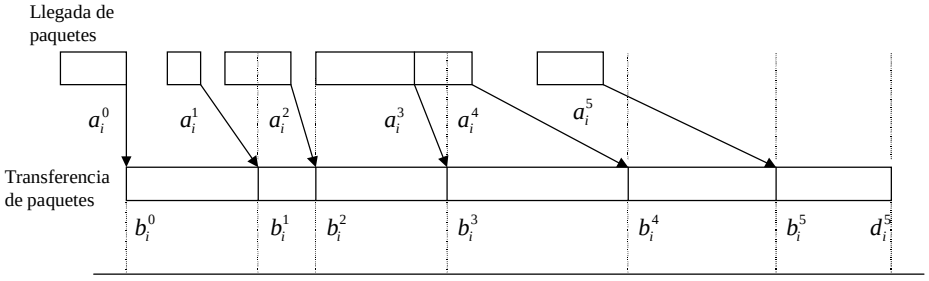

Figura 4.2. *Diagrama de tiempos de llegada y salida de paquetes en la disciplina de servicio GPS*

Una de las líneas de trabajo de GPS está centrada en el diseño de algoritmos de control de congestión basadas en realimentación para ser usados en redes de datagramas. Así, en este caso las fuentes muestréan constantemente el estado de la red para detectar síntomas de congestión y, cuando los detectan, las fuentes reducen la velocidad de envío para aliviar el efecto de la congestión. En el caso de que todas las fuentes empleen la misma cola FIFO para el envío de su tráfico, si una de las fuentes no reacciona al control de congestión, nos encontraremos con el hecho de que todas las sesiones acabarán sufriendo el problema de congestión. Para evitar este problema se propuso emplear una cola FIFO separada para cada sesión y atender las colas existentes en una política Round-Robin, como la propuesta (1) de Nagle [1] en la *Figura 4.1*. Este esquema se comporta mejor que un FIFO puro, pero en el caso que existan fuentes con diferentes tamaños de paquetes, puede ocurrir que las que generen paquetes más grandes acaben consiguiendo más ancho de banda que las que los generan más pequeños. Destacamos que en la disciplina de servicio PS, descrita antes, este problema no aparece porque las sesiones en espera reciben el mismo ancho de banda independientemente del tamaño de los paquetes.

El algoritmo GPS tiene por tanto una serie de interesantes propiedades para las redes de servicios integrados como ATM y, de hecho, se han propuesto muy diversos algoritmos PFQ (Packet Fair Queueing) [10] para aproximar GPS a las redes de conmutación de paquetes, aunque no demasiadas para las de tecnología ATM. El coste de implementación de los algoritmos PFQ está influido por dos importantes aspectos:

1) Cálculo de la función de tiempo virtual del sistema que es una medida dinámica del tamaño de servicio justo normalizado que debe recibir cada sesión. La mayor parte de algoritmos PFQ propuestos se mueven entre costes *O(1)* y *O(logN)* en cuanto a la complejidad de la función de tiempo virtual.

2) El coste del mantenimiento del orden relativo de los paquetes a través de su marca de tiempos en un mecanismo basado en colas con prioridad desde donde se transfieren los paquetes. La complejidad algorítmica de implementación de una cola con prioridad para *N* números arbitrarios es *O(logN).* Pueden conseguirse menores costes, pero son de difícil implementación en redes de alta velocidad. Las colas con prioridad en PFQ pueden basarse en el tiempo virtual de inicio, en el de finalización, o en ambos. Como cualquiera de estos dos tiempos crece monotónicamente con cada sesión, solo necesitamos considerar las cabeceras de cada sesión cuando el servidor está tomando el siguiente paquete a ser transferido. Así, el número de entidades en la cola con prioridad coincide con el número de sesiones activas.

Existen bastantes aproximaciones de algoritmos PFQ que procesan paquetes basados en GPS, pero todos usan un mecanismo similar de colas con prioridad, basado en la noción de una función de tiempo virtual. Las propuestas se diferencian en la elección de la función de tiempo virtual y en la selección de la política de paquetes.

Para poder aproximarse a GPS, los algoritmos PFQ mantienen una función de tiempo virtual *V(.),* un tiempo de inicio virtual $I_i(.)$ y un tiempo de finalización virtual $F_i(.)$ para cada una de las *i* sesiones. En el caso de las redes ATM es lógico representar el tiempo virtual en términos de la cantidad de células servidas.

$I_i(.)$ y $F_i(.)$ son actualizados cada vez que a una sesión *i* llega un paquete o está activa, o cada vez que un paquete de esa sesión *i* acaba su servicio. Podemos representar este funcionamiento con la siguientes expresiones,

$$I_i(t)\begin{cases} max(V(t), F_i(t-)) \\ F_i(t-) \end{cases} \tag{5}$$

$$F_i(t) = I_i(t) + \frac{L_i^k}{r_i} \tag{6}$$

donde $F_i(t-)$ es el tiempo de finalización virtual de la sesión *i* antes de la actualización, y L_i^k es la longitud del paquete de la cabecera de la cola de la sesión *i*.

Podemos intuir que *V(t)* es el tamaño de *tiempo de servicio normalizado justamente* que cada sesión debería recibir en el tiempo *t*. $I_i(t)$ representa el tamaño de *tiempo de servicio normalizado* que la sesión *i* ha recibido en el tiempo *t*. Como norma general, el objetivo de todos los algoritmos PFQ es el de minimizar las diferencias entre los tiempos virtuales *V(t)* e $I_i(t)$.

El papel que juega la función de tiempo virtual es el de reiniciar el valor del tiempo inicial virtual de la sesión cuando una sesión, que no estaba pendiente de servirse, pasa a estar pendiente de nuevo. Las funciones de tiempo virtual usadas en los diversos algoritmos PFQ existentes aportan diferencias entre su complejidad y su precisión[2].

En la mayor parte de los casos de PFQ, cuando el servidor está eligiendo el siguiente paquete para transmitir, los algoritmos optan, de entre todos los paquetes del sistema, por aquel que tenga el tiempo final virtual más pequeño, es decir, se aplica una política de selección de paquetes "primero los tiempos virtuales finales más pequeños" (SFF). Esta política de elección de paquetes suele dar lugar a límites de retardo casi idénticos a los del sistema fluido GPS, sin embargo, acaba dando lugar a mayores tiempos de servicio que GPS [10]. Desde luego, puede optarse por otras políticas de elección de paquetes como las descritas en [5,10,11], donde los algoritmos aplican la política SEFF (Smallest Eligible virtual Finish time First) optando, de entre todos los elegibles[3], por aquel con el menor tiempo de finalización.

Otra de las líneas importantes de GPS está en la forma de ofrecer servicios con garantía de retardo limitado en redes de conmutación de paquetes. El esquema PGPS (Packet Generalized Processor Sharing [2] responde al planteamiento (2) ilustrado en la *Figura 4.1*, con evolución hacia el (3), bajo el nombre de WFQ (Weighted Fair Queueing) que es el algoritmo PFQ más conocido. [2] demuestra que, empleando servidores GPS en los conmutadores, puede garantizarse un límite de retardo extremo-a-extremo a sesiones cuyo tráfico

[2] Una función de tiempo virtual se considera precisa si el algoritmo PFQ que la usa ofrece casi idéntico servicio que GPS.

[3] Un paquetes es elegible si su tiempo virtual de inicio es menor que el tiempo virtual actual.

es leaky bucket. La disciplina de servicio de PGPS se centra en el tratamiento igualitario de todos los usuarios y, a partir de estos planteamientos, han surgido posteriormente nuevas investigaciones que han dado lugar a variantes extendidas de PGPS.

Una de las variantes más interesantes de PGPS es descrita en [3,4] donde se emplea una disciplina de servicio también basada en topologías de redes arbitrarias de servidores GPS. Se destaca que para soportar servicios de tiempo real es básico resolver el problema de ofrecer garantía de funcionamiento a los diversos usuarios de una red de servicios integrados. Y este problema es especialmente difícil de resolver cuando aparecen las congestiones, que es cuando es importante el uso eficiente del ancho de banda de los enlaces de la red. En [3] se propone la combinación de control de admisión (CAC) leaky bucket y una disciplina de servicio de paquetes *work-conserving* en los nodos de la red para acomodar el retardo y el rendimiento de un amplio número de sesiones simultáneas. Mientras [3] analiza un sistema con un solo nodo, en [4] los mismos autores extienden el análisis a una red de topología arbitraria de servidores GPS, y relacionan los resultados GPS a redes en las cuales los nodos siguen la disciplina PGPS. En este segundo caso se parte de una red con una serie de valores fijados para los servidores, y un conjunto de sesiones sometidas a leaky bucket, y se intenta calcular cuál es el valor de retardo y de trabajo pendiente en el peor de los casos para cada una de las sesiones del conjunto. Para paquetes pequeños, como es el caso de ATM, el comportamiento de los sistemas PGPS y GPS es prácticamente idéntico.

Parekh en [2] estableció varias relaciones entre un sistema fluido GPS y su correspondiente sistema de paquetes WFQ que se pueden resumir en lo siguiente:

- En términos de retardo, un paquete acabará servicio en un sistema WFQ después que en el correspondiente sistema GPS, pero en no más tiempo de transmisión que el de un paquete del tamaño máximo.
- En términos del número total de bits servidos por cada sesión, un sistema WFQ no es peor que su correspondiente sistema GPS en más de un paquete del tamaño máximo.

Según lo anterior parece entenderse, y así se ha extendido la creencia, que la disciplina de paquetes WFQ y la disciplina fluida GPS ofrecen casi idéntico servicio diferenciándose únicamente en un paquete. Pero [5] se encarga de aclarar esta confusión ya que, aunque se puede probar que WFQ no cae en más de un paquete por debajo de GPS, sin embargo, WFQ sí que puede tener un comportamiento muy alejado del de GPS si lo expresamos en término de número de bits servidos por una sesión. Estas diferencias demuestran que WFQ puede acabar teniendo un comportamiento poco eficiente y [5] propone un nuevo y mejorado algoritmo de aproximación a GPS para la gestión de paquetes que llama Worst-case Fair Weighted Fair Queueing (WF^2Q y WF^2Q+) que ofrece casi idéntico servicio que GPS con una diferencia máxima de un paquete, y comparte el límite de retardo y las propiedades de justicia de GPS.

WF^2Q y WF^2Q+ son descritos en las referencias [10,11] y son presentados por sus autores como los algoritmos PFQ más precisos y, para ambos, se propone una mejora en [5], particularizada en ATM, en la que se consigue una mejora de la complejidad total, reduciendo la complejidad de las operaciones básicas de las colas además de los costes relacionados con las funciones de tiempo virtual. La idea de esta propuesta es la de alcanzar los siguientes objetivos:

- Soportar un gran número de VCIs con diferentes requerimientos de ancho de banda.
- Trabajar a altas velocidades del estilo de OC-3 o superiores a 155,52 Mbps.
- Mantener las propiedades de GPS en cuanto a límites de retardo, justicia y justicia en el peor de los casos.

Aunque WFQ es el algoritmo más conocido y reconocido como mecanismo de planificación ideal en términos de sus propiedades combinadas de límite de retardo y proporcionalidad de justicia, varios trabajos demuestran que la complejidad asintótica del algoritmo crece linealmente con el número de sesiones atendidas por el planificador, lo que limita su uso en aplicaciones de elevada velocidad. De este modo en [12] se proponen dos algoritmos de complejidad constante *O(1)* en gestión de las marcas de tiempos (pesos) y además, conservan los mismos valores de retardo extremo-a-extremo y tamaños de buffers de WFQ. El primer algoritmo FFQ (Frame-based Fair Queueing) emplea un mecanismo de tramas para recalibrar periódicamente una variable global rastreando la evolución de los trabajos en el sistema y limitando la injusticia en periodos cortos de tiempo determinados por el tiempo de las tramas. El segundo algoritmo SPFQ (Starting Potential-based Fair Queueing) realiza la recalibración en el límite de los paquetes.

Otra interesante alternativa es DRR (Deficit Round Robin) [7], que es un algoritmo que presenta una mejora de la implementación de WFQ. El esquema de gestión del buffer asume que cuando éste se llena, el

paquete de la cola más larga será tirado. DRR usa un algoritmo de encolado sofisticado y acaba consiguiendo un elevado grado de justicia.

RTA (Response-Time Analysis) [13] se propone como control de admisión de tráfico en tiempo real en redes ATM. RTA usa un sencillo mecanismo de prioridad de encolado que ofrece una clara separación entre ancho de banda y requerimientos de entrega. Esto da a RTA la posibilidad de mejorar la utilización del ancho de banda respecto a WFQ. Además, se compara RTA con CND (Calculus of Network Delays) que también emplea priorización en las colas.

Investigaciones posteriores se centran en la propuesta de entornos donde la computación de los límites de retardo y ancho de banda estén más controlados, al contrario de las propuestas GPS que se basan en garantías de QoS determinísticas que pueden ser consideradas como más conservadoras debido a los relajados límites que se aplican y que conducen a limitaciones de capacidad. De este modo se proponen en [14] varios algoritmos de CAC basados en sistemas con retardo y ancho de banda desacoplados.

Por último, RFQ (Rainbow Fair Queueing) [15] parte de los esquemas previos [16] como CSFQ que no emplean *per-flow* en el mecanismo justo para el reparto del ancho de banda entre los diferentes flujos. RFQ divide cada flujo en un conjunto de capas basadas en velocidades. Los paquetes de cada flujo son "marcados con un color", de forma que los routers mantienen un umbral de color y se encargan de tirar todas las capas cuyo color excede del umbral.

4.4. PROPUESTA QPWFQ PARA TAP

A la vista del estudio de las propuestas ya comentadas podemos decir que WFQ, también denominado PGPS (Packet-by-packet Generalized Processor Sharing), aporta interesantes prestaciones en cuanto a retardo y justicia [3,4], sin embargo, su coste computacional y complejidad de implementación han impedido su implantación.

VirtualClock y SCFQ proponen un mecanismo de planificación de paquetes más simple que WFQ, aunque las tres propuestas usan la longitud de paquetes, el peso y el tiempo virtual como parámetros para la planificación de los paquetes que llegan a las colas de entrada de los conmutadores ATM.

Delay-EDD, Jitter-EDD y HOL-EDD se basan en la asignación de marcas de tiempos de servicio a las colas o a los paquetes que son enviados en función de esos tiempos de servicio.

Todos estos algoritmos de planificación de colas se basan en marcas de tiempos que se asignan a las colas o a los propios paquetes garantizando límites en los retardos mediante el tratamiento aislado del tráfico. Sin embargo, la introducción de las marcas de tiempos provoca los excesivos costes computacionales ya comentados que puede determinarse en $O(\log_2 N)$ donde N es el número de VPIs/VCIs a la espera en las colas. El coste logarítmico es aceptable cuando el número de conexiones no es elevado, pero cuando se multiplexan varios miles de conexiones en un único enlace ATM este coste se considera excesivo. Han aparecido propuestas con costes $O(1)$ pero, por una causa u otra, se ajustan mal a las redes de tecnología ATM. Otro inconveniente de los algoritmos basados en marcas de tiempos está en uno de los más importantes parámetros de QoS en las redes de tecnología ATM, que es la variabilidad de los retardos (*jitter*). Esto ha llevado a varios investigadores a indicar lo inadecuado de estos algoritmos desde el punto de vista de la compartición de recursos.

No obstante, la literatura presenta diversas variantes mejoradas sobre las propuestas anteriores. Así, por ejemplo, en [17] se presenta una variación de SCFQ con coste constante $O(1)$. En nuestro caso queremos centrarnos en una de estas propuestas, QLWFQ (Queue Length based WFQ) descrita por Ohba en [18] y donde se presenta un algoritmo *work-conserving* de coste constante $O(1)$ para la planificación de colas *per-VC* en redes de conmutación de paquetes de longitud fija como ATM. QLWFQ es compatible con FQ [1] y FIFO y se basa en la asignación de pesos a las colas y en el establecimiento de créditos para determinar el orden de atención de las células ATM en función de la comparación de la longitud de cada cola con su peso particular.

El algoritmo compara -a la llegada y partida de las células- la longitud de las colas con los pesos que cada una tiene asignados. QLWFQ es en realidad una simplificación de WRR (Weighted Round Robin) [19] consiguiendo un mejor equilibrio entre aislamiento de tráfico y compartición de recursos que los algoritmos basados en marcas de tiempos. Ohba demuestra unos aceptables límites de retardo e interesantes índices de justicia a través de simulaciones de escenarios de elevada y de baja carga de tráfico.

En nuestro caso proponemos el algoritmo QPWFQ (Queue PDU Weighted Fair Queueing), inspirado en QLWFQ, para conseguir aportar un tratamiento justo a las PDU que llegan a los conmutadores AcTMs. Dado

que debemos dar un tratamiento privilegiado a las PDU pertenecientes a las conexiones con GoS, el mecanismo de pesos de colas es de suma utilidad para nosotros. Además, nos encontramos con otra particularidad no tratada por ninguno de los algoritmos estudiados que es el hecho de tener que dar respuesta a las retransmisiones de PDUs congestionadas. Esto nos va a determinar el tener que incluir en el algoritmo QPWFQ un tratamiento prioritario para las retransmisiones.

A continuación vamos a describir el funcionamiento general del algoritmo QPWFQ y, a un tiempo, comentaremos el mecanismo aplicado a las colas de entrada de los conmutadores AcTMs para demostrar la justicia del mismo, así como su sencillez de implementación que acaba resultando en un coste constante *O(1)*. Igualmente, veremos que QPWFQ procesa, tanto células como PDU, y describiremos la sencilla técnica usada para mantener las características *work conserving* de WFQ, así como la posibilidad de tratamiento prioritario de las solicitudes de retransmisiones de PDUs pertenecientes a las conexiones garantizadas.

La *Figura 4.3* ilustra el comportamiento de QPWFQ que es gestionado por el agente programable WFQA[4]. En esta figura podemos observar cómo las fuentes de tráfico llegan al agente WFQA a través del agente CoSA que se encarga de aplicar las características de tráfico negociadas para las conexiones. Supondremos que el tráfico de llegada es de células nativas ATM (aunque igualmente podrían ser PDU si suponemos estar en el conmutador de inicio de la conexión). Todo el tráfico es controlado por CoSA que se encarga de detectar las conexiones privilegiadas y comenzar el reensamblado de las células pertenecientes a cada una de ellas. CoSA sirve las PDU o las células a su correspondiente cola de llegada que es donde después serán tratadas por el algoritmo QPWFQ.

Como puede observarse en la *Figura 4.3*, tenemos tres colas de espera, cada una de ellas con su correspondiente peso *p*. En nuestro caso la Cola 1 tiene peso 3, la cola 2 peso 2 y la cola 3 peso 1. Las posiciones de cada cola pueden almacenar, tanto PDU como células para ser procesadas hasta el buffer del conmutador. Cuando una PDU, o una célula, llega a una cola, la longitud *l* de esta cola es incrementada en una unidad y es introducido el número de cola en la *cola de turnos* siempre que $l<=p$.

Cuando la cabecera de una cola de espera es servida, la longitud de esta cola es decrementada en una unidad y se comprueba si $l>=p$ para seguir atendiéndola, siempre y cuando en las restantes colas no se cumpla $l<=p$. De este modo se garantiza la característica *work-conserving* (el servicio no se detiene mientras exista tráfico que procesar), que permite atender colas con tráfico continuado siempre y cuando en las restantes colas no exista tráfico.

En el escenario de la *Figura 4.3* hemos supuesto que en el instante 0 en la cola 1 había una PDU, la cola 2 estaba vacía y en la cola 3 existía otra PDU. Así, la longitud de la cola 1 se incrementa en 1, y en la *cola de turnos* se ha introducido el crédito 1 (suponiendo que atendemos las colas circularmente y de forma ascendente) ya que su peso es 3 y la longitud de la cola en ese instante es 1 ($l<=p$). Posteriormente es tratada la cola 3, su longitud es incrementada en uno, y es anotado su turno en la *cola de turnos* ya que su peso es 1, lo mismo que su longitud ($l<=p$). Si suponemos mantenernos en esta situación sin servir las cabeceras de las colas 1 y 3 a la salida, y en ese momento llegan tres nuevas unidades a la cola 1, éstas serán atendidas de forma continuada por la característica *work-conserving*, ya que a las colas 2 y 3 no llega tráfico. De este modo se atienden dos nuevas entradas de la cola 1 pero al ser atendida la tercera llega una célula a la cola 2 que pone su longitud a 1 y es pasado su turno a la *cola de turnos* ya que $l=1 <= p=2$.

El procedimiento de salida de las células o PDU de las colas de espera funciona de la siguiente forma. La cabecera de la cola de turnos indica la cola de espera que va a ser atendida en cada momento. Volviendo a la *Figura 4.3*, se observa cómo es el turno de atender la cola de espera 1, entonces lo que se hace es enviar la cabecera de la cola 1 hasta el buffer, a continuación se decrementa en 1 la longitud de la cola 1, se elimina la cabecera de la *cola de turnos* y de la cola 1, para pasar a comprobar primero si $l<=p$ y después si $l>=p$.

En el escenario descrito no hemos considerado aún la posibilidad de llegada a las colas de espera de PDU pertenecientes a solicitudes de retransmisión de conexiones privilegiadas. Cuando esto ocurre, el agente WFQA realiza un tratamiento privilegiado de estas retransmisiones para darles prioridad en el servicio, tanto con respecto al resto de PDU de esa misma conexión como con respecto al resto de conexiones. Es decir, cuando a una cola de espera llega una PDU retransmitida se prioriza con respecto a las PDU que ya estuvieran en la cola de espera de su propia conexión y, además, también se priorizará su cola de espera (en la *cola de turnos*) con respecto al resto de colas de espera. Para implementar esta posibilidad recurrimos también a un sencillo mecanismo que mantenga el coste constante ya comentado y sin necesidad de usar excesivas variables para su implementación.

[4] El sistema multiagente es descrito en el Capítulo 6.

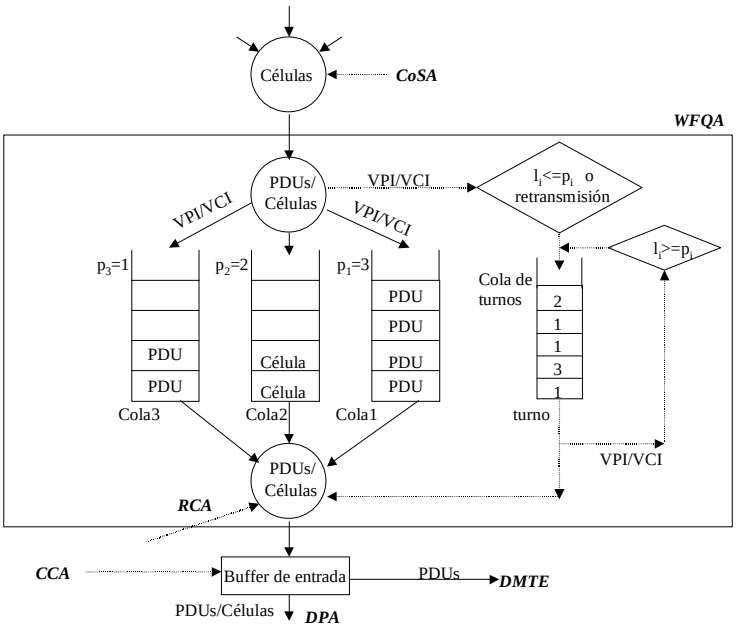

Figura 4.3. *Esquema de planificación del algoritmo QPWFQ*

El WFQA tendrá conocimiento de la inminente llegada de una PDU retransmitida a través de la comunicación del agente RCA que le indica el índice VPI/VCI/PDUid/Port. Esta comunicación activa una nueva variable asociada a cada cola para saber que va a llegar una retransmisión. Cuando llega la PDU es introducida en su correspondiente cola y es apuntada para garantizar el acceso directo. De este modo, las colas de espera pueden servir células y PDU, no sólo desde las cabeceras, sino también desde los punteros que identifican las PDU retransmitidas. Así, las colas FIFO, implementadas con punteros, permitirán priorizar las PDU retransmitidas que, a medida que van llegando, son insertadas, no al final de las colas, sino al principio de las mismas con la intención de garantizar el coste constante.

Pero para poder realizar la priorización también será necesario actuar en la *cola de turnos* para priorizar la cola a la que acaba de llegar la PDU retransmitida. Para poder realizar esta última operación es necesario comprobar si se trata de una retransmisión para poder acceder a la cola de turnos. Cuando se cumple esta condición el número de cola es priorizado en la *cola de turnos* siendo colocada en la cabeza de la misma.

Aunque las colas pueden contener células nativas ATM y PDU privilegiadas y, aparentemente esto puede conducir a situaciones injustas en las células respecto a las PDU, la estrategia aplicada, basada en los pesos y las longitudes de las colas, permite que no se produzcan situaciones de inanición de paquetes pequeños respecto a los de mayor tamaño.

En capítulos posteriores es descrito detalladamente el algoritmo QPWFQ así como su funcionamiento sobre varios escenarios de simulación.

4.5. CONCLUSIONES

En la actualidad, el control de congestión se delega en protocolos que las resuelven mediante retransmisiones extremo-extremo. Esta es una técnica sencilla de implementar a altas velocidades que también simplifica los conmutadores, pero toda la red se ve sobrecargada con las retransmisiones y tampoco aporta protección contra fuentes de tráfico egoístas.

Hemos revisado diferentes esquemas de planificación de paquetes inspirados en el método de tráfico continuo GPS, y hemos podido identificar sus ventajas e inconvenientes. Los esquemas de ancho de banda justo protegen a las fuentes bienintencionadas de las que no se comportan adecuadamente, y permiten un amplio y variado conjunto de mecanismos de control de congestión extremo-extremo. La clave de los algoritmos descritos en la literatura es su propia dificultad de implementación debido a la complejidad de mantener las colas de prioridad y la computación de la función de tiempo virtual. Esta complejidad crece con el incremento del número de sesiones que son procesadas pero, en nuestro caso, esta cuestión se ve amortiguada por el hecho de que las conexiones privilegiadas que van a disponer de GoS constituyen un número reducido y controlado.

Nuestra propuesta está basada en QLWFQ por las atractivas ventajas que aporta este algoritmo. No obstante, las características propias de nuestra arquitectura nos llevan a proponer el algoritmo QPWFQ para soportar los siguientes aspectos básicos inherentes a TAP:

- En nuestro caso, la unidad de procesamiento en las colas de entrada a los conmutadores AcTMs son, tanto PDUs, como células nativas ATM. Este aspecto en realidad no afecta a las características de planificación, ya que aprovechamos los tiempos de espera en las colas para realizar el reensamblado de las células ATM y obtener las PDU que son la base de funcionamiento del protocolo TAP.
- En segundo lugar, las colas de espera van a requerir mayores tamaño de memoria por la significativa diferencia entre el tamaño de las PDU y las células ATM nativas.
- En tercer lugar, una de las prestaciones más importantes de TAP es el mecanismo de retransmisiones de PDU que no es soportado en QLWFQ. Para nuestra propuesta es vital la búsqueda del equilibrio entre la justicia en el procesamiento del tráfico, y la atención privilegiada de las solicitudes de retransmisión de PDU. Por tanto, QPWFQ mantiene los índices de justicia y el coste computacional constante soportando la priorización de las retransmisiones.
- Por último, y como aspecto más destacable, ante la posibilidad de soportar la solicitud por parte de los usuarios, o de protocolos de control de flujo de cambios dinámicos en los pesos de las colas, QPWFQ es parte de un agente programable de nuestra arquitectura. Este planteamiento implica el establecimiento y control de las comunicaciones del agente WFQA con el resto de agentes del sistema multiagente propuesto para el soporte de TAP.

REFERENCIAS

[1] J. Nagle, "On packet switches with infinite storage," *IEEE Transactions on Communications*, pp. 435-438, (1987).

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proceedings of ACM SIGCOMM'89*, pp. 3-12, (1989).

[3] Abhay Parekh and Robert G. Gallager, "A generalized processor sharing approach to flow control - the single node case," *ACM/IEEE Transactions Network*, (1993).

[4] Abhay Parekh and Robert G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the multiple node case," *IEEE/ACM Transactions on Networking*, pp. 137-150, (1994).

[5] J. C. R. Bennet and H. Zhang WF^2Q: Worst-case Fair Weighted Fair Queueing," *Proccedings of IEEE INFOCOM'96*, pp. 120-128, (1996).

[6] S. J. Golestani, "A Self-Clocked Fair Queueing scheme for broadband applications," *Procs. IEEE 13th INFOCOM*," pp. 636-646, (June 1994).

[7] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *Proceedings SIGCOMM'95*, pp. 231-243, (1995).

[8] D. Lin and R. Morris, "Dynamics of random early detection," *Proceedings of ACM SIGCOMM'97*, pp. 127-137, (1997).

[9] S. Floyd and V. Jacobson, "Random early detection for congestion avoidance," *IEEE/ACM Transactions on Networking*, pp. 397-413, (1993).

[10] Jon C. R. Bennett, D. C. Stephens and Hui Zhang, "High speed, scalable, and accurate implementation of packet fair queueing algorithms in ATM networks," *Procs. International Conference on Network Protocols*, pp. 7-14, (Oct. 1997).

[11] Jon C. R. Bennet and Hui Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*," pp, 675-689, (Oct. 1997).

[12] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Transactions on Networking*, pp. 175-185, (April 1998).

[13] H. Hansson and M. Sjodin, "Response time guarantees for ATM-networked control systems," *Procs. IEEE International Workshop on Factory Communication Systems*, pp. 213-222, (Oct. 1997).

[14] R. Szabo, P. Barta, F. Nemeth, J. Biro and C.-G. Perntz, "Call admission control in generalized processor sharing (GPS) schedulers using non-rate proportional weighting of sessions," *Procs. IEEE 19th INFOCOM*, pp. 1243-1252, (March 2000).

[15] Zhiruo Cao, Zheng Wang and E. Zegura, "Rainbow fair queueing: fair bandwidth sharing without per-flow state," *Procs. 19th INFOCOM*, pp. 922-931, (March 2000).

[16] Ion Stoica, Scott Shenker and Hui Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocation in High Speed Netwotks,", *Proceedings of SIGCOMM'98*, (1.998).

[17] J. L. Lexford, A. G. Greenberg, and F. G. Bonomi, "Hardware-Efficient Fair Queueing Architectures for High-Speed Networks," *Proceedings IEEE INFOCOM*, pp.638-646, (1996).

[18] Yoshihiro Ohba, "QLWFQ: A Queue Length Based Weighted Fair Queueing Algorithm in ATM Networks", *INFOCOM'97, Proceedings IEEE*, pp. 566-575 vol.2 (1997).

[19] M. Katavenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a general-purpose ATM switch chip," *IEEE JSAC*, pp. 1265-1279, (1991).

[20] N. R. Figueira and J. Pasquale, "An Upper Bound on Delay for the VirtualClock Service discipline," *IEEE/ACM Transactions Networking*, pp. 399-408, (1995).

[21] H. Zhang and S. Keshav, "Comparison of Rate-Based Service Disciplines," *Proceedings ACM SIGCOMM*, pp. 113-121, (1991)

[22] M. Vishnu and J. W. Mark, "HOL-EDD: A flexible Service Scheduling Scheme for ATM Networks," *Proceedings IEEE INFOCOM*, pp. 647-654, (1.996).

CAPÍTULO 5

MECANISMOS DE CONTROL DE CONGESTIÓN

5.1. INTRODUCCIÓN

Sabemos que la CoS UBR fue diseñada y propuesta para las aplicaciones de datos sin especiales requerimientos de retardo, que no son sensibles a las pérdidas de células y que aprovechan la capacidad disponible en la red generando tráfico a ráfagas. De este modo, no están controladas por la función CAC y tampoco se ven sometidas a políticas de rendimiento. Si aparecen congestiones, las células pueden perderse y las fuentes no se ven obligadas a reducir su velocidad de transmisión. Por esto, son las aplicaciones las que se encargan de aplicar sus propios mecanismos de retransmisión y de recuperación de las pérdidas, generalmente a través de la ventana de control de flujo de TCP.

Una CoS alternativa a UBR es, como también sabemos, ABR que aporta su propio mecanismo de control de congestión basado en velocidades en un bucle cerrado, en el que el uso de las células RM juega un importante papel. Este mecanismo consiste en que las fuentes de entrada se encargan de ajustar sus velocidades a los niveles de congestión de las conexiones en cada momento. En esta línea, se han propuesto múltiples esquemas de control de congestión basados en las velocidades de las fuentes, que se diferencian básicamente en la forma en que los conmutadores determinan la congestión y en la forma en que las fuentes condicionan sus velocidades de transmisión a las situaciones de congestión. Los propuestas iniciales para aplicar esos mecanismos de control de congestión en ABR aportadas por el ATM Forum [1,2,3,24] buscan la justicia de las fuentes sin aplicar ninguna técnica de encolado *per-VC* que, posteriormente, han acabado popularizándose una vez que se ha demostrado la ventaja del tratamiento diferenciado de los flujos en colas separadas.

En suma, ABR y UBR son las dos clases de servicio diseñadas por el ATM Forum [4] para soportar las aplicaciones que generan datos sin requerimientos de tiempo real. Aunque ABR aporta mejores características de QoS, UBR es más simple en su implementación con una complejidad y coste menores que ABR. Por esto UBR es una interesante alternativa a ABR y, sea una u otra la CoS usada, hemos de buscar soluciones para controlar las situaciones de congestión y para eliminar los problemas de injusticia que puedan producirse en las fuentes. La existencia de tráfico a ráfagas provoca que las redes ATM acaben alcanzando situaciones de sobrecarga y colapso más a menudo de lo esperado.

Recordamos que UBR y ABR son las CoS destinatarias de la arquitectura que hemos diseñado y destacamos que, tanto ABR como UBR, han sido las propuestas para soportar las aplicaciones de datos (principal objetivo de la arquitectura TAP) en las redes ATM. Precisamente, TCP es de los protocolos de la capa de transporte más usados en redes de datos y se han realizado numerosos esfuerzos para su soporte sobre las redes ATM. Es de suma importancia, por tanto, dar respuesta a múltiples requerimientos de este tipo de tráfico best-effort, ya que en la actualidad es el más abundante en las redes.

En el Capítulo 3 se comenta que las principales investigaciones para conseguir fiabilidad están inspiradas en mecanismos de generación de código redundante (como CRC o FEC), los cuales introducen importante overhead generando sobrecargas en la red y afectando al throughput negativamente. Sabemos también que estos métodos redundantes solventan el problema de la recuperación de células corruptas y perdidas, pero no pueden solventar otros indeseables, y no menos frecuentes, problemas como son la congestión y la

fragmentación[1] de paquetes en los conmutadores, debidas al exceso de tráfico, a problemas en los buffers, etc. Todos estos problemas acaban afectando al rendimiento de la red y degradando el goodput en la misma.

Los esquemas de control de congestión más conocidos son RCD (Random Cell Discard), PPD (Partial Packet Discard) [5] y EPD (Early Packet Discard) [1,6]. Avanzamos ya que el esquema EPD se ha demostrado como una solución interesante para conseguir óptimo funcionamiento desde el punto de vista de la justicia y utilización de los enlaces. Sin embargo, existen otros esquemas con idénticos objetivos que son maximizar el throughput y la justicia, mientras se minimiza el retardo en las redes ATM. Algunos de estos mecanismos son: ESPD (Early Selective Packet Discard) [3], FBA (Fair Buffer Allocation) y RED (Random Early Detection) [8,9].

En los trabajos previos de nuestras investigaciones [10] implementábamos PPD para aliviar el efecto de las congestiones y de la fragmentación de paquetes. Posteriormente hemos soportado y modificado una variación de EPD en TAP de forma que, cuando una de las PDU llega al buffer, esperamos a la célula final de cada PDU para aplicar EPD en los conmutadores AcTMs, desechando aquellas PDU que provoquen congestión en el buffer y solicitando su retransmisión mediante NACKs al conmutador previo. Las PDU que no provocan congestión en el buffer son enviadas a los puertos de salida de forma completa aplicando VC merge para aliviar también los problemas de interleaving[2].

Cuando se llena el buffer de un conmutador congestionado, las células que llegan serán descartadas y, con una elevada probabilidad, muchos paquetes perderán células y mientras otras células seguirán su transmisión a través de la red, provocando el indeseable fenómeno que [11] de la fragmentación[3] de paquetes. Al evitar la fragmentación estamos optimizando el ancho de banda de los enlaces de la red que no se ven sobrecargados con las retransmisiones extremo-extremo, ni con la transmisión de PDU completas que pueden estar corruptas por la pérdida de una sola de sus células.

En este capítulo vamos a revisar diferentes técnicas para eliminar o aliviar los problemas de congestión de buffers y fragmentación de paquetes que provocan la degradación progresiva del throughput de la red. Varios de estos mecanismos buscan también un cierto punto de equilibro entre el índice de justicia que son capaces de aportar y el googput que se consigue en la red. Es decir, se busca no sacrificar la justicia de las fuentes por incrementar el throughput en la red, teniendo en cuenta que el índice de justicia puede ser calculado con la siguiente expresión [12],

$$Ij = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\sum_{i=1}^{n} x_i^2}$$

donde x_i es el througput efectivo de la *i-esima* fuente, y n es el número de fuentes.

En el caso de la arquitectura TAP la justicia es garantizada por la estrategia planteada en el capítulo anterior donde se presenta el algoritmo QPWFQ. Por esto, el buffer que usamos, y donde aplicaremos una variante del protocolo EPD, se propone como un mecanismo para evitar la fragmentación de las PDU y también como el punto en que realizamos la detección de congestiones y solicitud de retransmisiones. Para evitar el problema de la fragmentación proponemos la aplicación de técnicas VC-merging[4] que también van a ser descritas en el presente capítulo. El buffer es gobernado por un agente programable que permite al operador de la red elegir el algoritmo a aplicar en la gestión del buffer.

5.2. PPD (PARTIAL PACKET DISCARD)

PPD fue el esquema propuesto por G. Armitage y K. Adams [5] para amortiguar el efecto negativo de la fragmentación de paquetes, lo que supone mejorar sustancialmente el throughput conseguido por el esquema original RCD (Random Cell Discard) basado en desechar las células aleatoriamente. PPD tira las células cuando, al llegar un paquete, se llena el buffer, y continua tirando las siguientes células del mismo paquete.

[1] La fragmentación de los paquetes se produce al perderse células de forma incontrolada provocando la inconsistencia completa de los paquetes (PDUs) a los que pertenecen las células perdidas.

[2] El interleaving se produce al intercalarse en un mismo enlace las células pertenecientes a conexiones diferentes.

[3] Cuando se tira alguna célula de un paquete, éste no podrá ser reconstruido por el destinatario, lo que requerirá la retransmisión completa del paquetes extremo-extremo, afectando negativamente al goodput de la red.

[4] VC merge es una de las técnicas propuestas para evitar el problema del interleaving.

Este comportamiento de PPD conduce a tirar las colas de cada paquete. No obstante, PPD aporta un mejor comportamiento a las conexiones TCP aplicando un mecanismo selectivo (en lugar de aleatorio como RCD) en las células que son tiradas por la red.

En el caso del envío de paquetes (PDU) a través de las redes ATM supone que éstos sean segmentados en células de 53 octetos que son las que realmente viajan por la red. Por tanto, todas las células de un paquete deben llegar íntegras al destinatario para que el paquete llegue sin problemas. Cuando una sola célula se pierde, el paquete completo acabará también perdiendo su integridad por lo que será descartado. Es por tanto evidente el riesgo de afectar a muchos paquetes aunque sólo se pierdan unas cuantas células si se desechan las células de forma aleatoria. Pero además, ocurre que el resto de las células de un paquete que acaba de perder una célula continúan adelante en dirección al destino, pudiendo volver a provocar nuevas congestiones que acaben afectando a otras conexiones que pueden también experimentar pérdidas. Con lo cual, células que pertenecen a paquetes que ya se sabe que son corruptos pueden colaborar a la congestión de nuevos conmutadores. Este es el citado fenómeno que se definió en [11] como fragmentación, para el que en [5] se propuso el algoritmo PPD presentado en la *Figura 5.1*.

En este algoritmo la *lista-descartes* registra una relación de VPI/VCIs que quedan marcados cuando ya se les ha tirado alguna célula en el paquete que se está procesando para cada VPI/VCI. Con EOM se determina cuándo se llega a la célula final de un paquete. Cada vez que se acaba de procesar un paquete que ha perdido alguna célula se actualiza la *lista-tirados* eliminando el VPI/VCI de esta lista para comenzar el siguiente paquete de esa conexión sin tenerlo marcado con células descartadas.

```
Mientras una célula está llegando a un conmutador;
    Si el VPI/VCI de la célula pertenece a lista-descartes
        tirar la célula
        Si la célula es una célula EOM
            borra el VPI/VCI de la célula de la lista-descartes
        Fsi
    en otro caso
        Si el buffer está lleno
            tirar la célula
            añadir el VPI/VCI de la célula a la lista-descartes
        en otro caso
            aceptar la célula en el buffer
        Fsi
    Fsi
FMientras
```

Figura 5.1. *Algoritmo PPD*

Como puede observarse en el algoritmo, PPD tirará una célula cuando se llena el buffer pero, además, como se registra el VPI/VCI de todas las células tiradas en la *lista-descartes*, también se tirarán todas las células que pertenezcan al mismo paquete y que siguen a la que ya ha sido descartada. De este modo, no sólo se tiran las partes finales de los paquetes, sino que se pueden tirar paquetes completos. Aunque no se elimina completamente el problema de la fragmentación, sí se reduce de una forma importante el número de las células inservibles.

5.3. EPD (EARLY PACKET DISCARD)

EPD es una técnica de gestión de buffers implementada para asegurar elevado throuhgput extremo-extremo a las aplicaciones de datos a ráfagas durante los periodos de sobrecarga. EPD aporta mejoras a PPD, con la intención de garantizar que se tiran paquetes enteros en lugar de partes de paquetes como hace PPD. EPD fue introducido en [6] para tirar las células BOM (Begin Of Message) y las siguientes células del paquete a que pertenecen las cabeceras, una vez que se ha superado un determinado umbral de llenado del buffer. EPD tirará los paquetes completos antes de que el buffer acabe llenándose, con la intención de que los paquetes que puedan ser susceptibles de perder alguna célula, y por tanto su integridad, no sean transmitidos por la red.

En resumen, cuando un buffer se llena, en lugar de tirar las células pertenecientes a diferentes conexiones que generan paquetes, o PDU en nuestro caso, se propuso en [11] EPD para tirar los paquetes completos antes de que el buffer se llene. De este modo se evita el problema de la fragmentación y se consigue que no sigan adelante las transmisiones de los paquetes corrompidos por la pérdida de células aleatorias. En realidad,

podemos entender [13,14] que EPD es un algoritmo para el descarte de paquetes, que puede ser aplicado a protocolos basados en paquetes soportados sobre ATM como TCP, UDP o IPX. En nuestro caso, lo emplearemos para soportar la posibilidad de tirar PDUs generadas por nuestro protocolo TAP ATM nativo. Varias investigaciones [7,14,15,16] han demostrado el rendimiento de TCP sobre ABR y UBR en términos de throughput en la red y justicia en las fuentes de tráfico. Se ha demostrado también que EPD, aplicado a tráfico TCP sobre UBR, no consigue buenos índices de justicia y, sobre todo, en redes muy congestionadas. Sin embargo, mejorando EPD con mecanismos *per-VC accounting* y *per-VC queueing* se consigue un buen comportamiento en cuanto a justicia en el tratamiento de los VC.

A continuación vamos a comprobar la simplicidad del algoritmo para comprender que su coste algorítmico es asumible desde el punto de vista de la justicia y goodput conseguidos. Para soportar EPD se establece en cada conmutador un valor umbral que marca el punto de llenado del buffer en que, al ser alcanzado, se supone que no se aceptará la entrada de ningún otro paquete. Es decir, cuando llega la primera célula de un paquete a un buffer que está lleno hasta, o por encima de, su valor umbral, esa célula es desechada, y también todas las demás células pertenecientes a ese paquete, aunque la cola descienda inmediatamente su nivel de llenado por debajo del umbral. Por otro lado, las células de un paquete cuya primera célula llegó antes de que el buffer alcanzase el umbral no serán tiradas a no ser que el buffer acabe llenándose. En suma, cuando la primera célula de un paquete llega en un margen de llenado elevado del buffer, es descartada junto a todas las demás que pertenecen a su mismo paquete, ante el riesgo de provocarse la fragmentación del paquete.

Lo importante está en elegir el valor del umbral para que sea capaz de evitar el rebosamiento del buffer que conduce a la fragmentación. Pero además, es necesario buscar el punto de equilibrio al establecerlo para que al intentar evitar la fragmentación no acabemos provocando demasiadas retransmisiones de paquetes desechados completamente. Es decir, si se establece el umbral demasiado alto, el mecanismo no surtirá ningún efecto y, si el umbral se fija muy bajo, el porcentaje de paquetes tirados será demasiado elevado degradando también el goodput.

La *Figura 5.2* ilustra el comportamiento del algoritmo EPD cuyos detalles de funcionamiento pueden ser ampliados en la referencia [11]. Además, la *Figura 5.3*, presenta el algoritmo EPD implementado en [13,14] donde se demuestra que el ancho de banda es mejor aprovechado con EPD aplicado tanto a ABR como a UBR. Hemos adaptado la *Figura 5.3* a nuestro escenario concreto y podemos observar la llegada de las PDU al buffer del conmutador, en el que hemos definido el *tamaño del umbral*, el *tamaño actual de la cola* (*TAC*) y la *capacidad máxima del buffer* (*CMB*). En la parte superior de la *Figura 5.3* (caso a)), puede observarse cómo al buffer ha llegado la PDU 7, que no cabe completa en el buffer. Sin embargo, como a la llegada de la cabecera (primera célula) de esta PDU el valor actual de la cola está por debajo del umbral (δ=*Umbral-TAC>0*), la PDU es aceptada en el buffer, y el tamaño de la cola se ve incrementado en el volumen de la PDU. En el escenario b), al crecer el tamaño de *TAC,* éste se acaba igualando con el valor del umbral, lo que provoca que, a la llegada de la primera célula de la PDU 8 (δ=*Umbral-TAC=0*), ésta no pueda ser admitida aunque aún queda espacio en el buffer del conmutador. De este modo la PDU 8 es tirada por la red ante el riesgo de experimentar la fragmentación debida al probable rebosamiento del buffer. Si la PDU 8 no fuese descartada, sus células finales no cabrían en el buffer, mientras las primeras seguirán adelante provocando la fragmentación de la PDU.

```
TAC=Tamaño_actual_de_la_cola
CMB=Capacidad_máxima_del_buffer_del_conmutador
Mientras una célula está llegando a un conmutador;
    Si la célula es la primera célula de un paquete
        Si TAC >= Umbral
            tirar la célula
        En otro caso
            aceptar la célula en la cola FIFO
        FSi
    En otro caso
        Si ya se ha tirado alguna célula de este paquete
            tirar la célula
        En otro caso
            Si TAC >= CMB
                tirar la célula
            En otro caso
                aceptar la célula en cola FIFO
            Fsi
        Fsi
    Fsi
FMientras
```

Figura 5.2. *Algoritmo EPD*

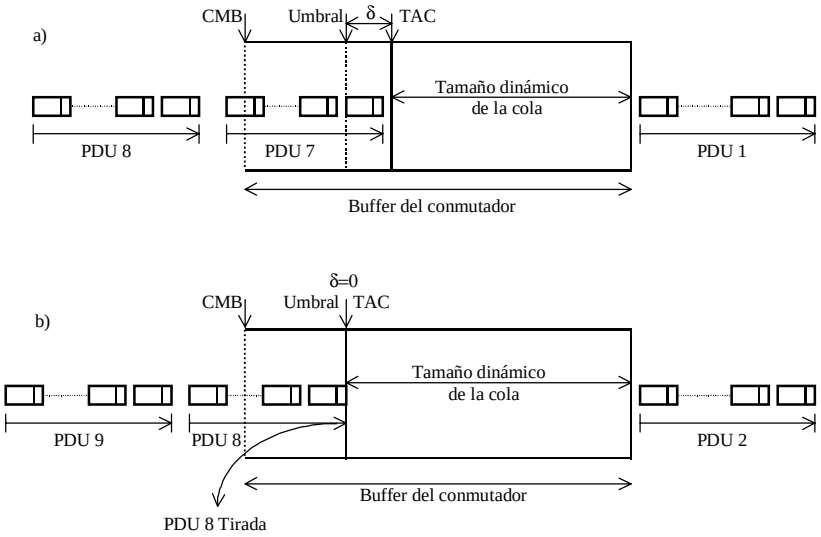

Figura 5.3. *Representación gráfica de EPD procesando PDUs AAL-5*

Investigaciones como [13,14] demuestran que el índice de justicia en ABR y UBR con control de congestión basado en velocidades, puede ser mejorado aplicando esquemas EPD en colas *per-VC*. Los autores de estas investigaciones han demostrado también en trabajos previos que TCP sobre ABR tiene mejor comportamiento que sobre UBR, necesitando incluso menos requerimientos en cuanto a hardware.

El algoritmo de EPD puede tener otros planteamientos o versiones, aunque el problema general es el mismo. De hecho, en la sección 5.7 explicaremos nuestra propia variante adaptada a ATM, que es la que implementamos en nuestra arquitectura. Usamos la condición de EOM (End Of Message) que es la delimitación de PDU que soportamos en nuestra propuesta nativa de EAAL-5.

No obstante, la aplicación de EPD al tráfico UBR no permite garantizar el tratamiento justo de las fuentes de tráfico y se ha comprobado [13,14] que el tráfico de varias fuentes acaba teniendo un comportamiento bastante injusto, lo que justifica la utilización de técnicas *per-VC* para intentar evitar el efecto de unas fuentes sobre otras. Para ello se han propuesto dos variantes del algoritmo EPD original [17] para incorporar asignación justa de buffer, *per-VC queueing* y *per-VC accounting*. Estos dos mecanismos intentan mejorar la técnica aleatoria de tirado de paquetes que provoca la asignación de ancho de banda injusta entre fuentes que se disputan el buffer.

El mecanismo *per-VC accounting* se basa en el uso de una nueva condición para tirar las PDU, de forma que, además de cumplir que *TAC >= umbral*, se debe cumplir que $TAC_i >= UM$ para cada fuente F_i . En este caso, TAC_i es el número de células pertenecientes a la fuente F_i . *UM* representa el porcentaje del *umbral total* que corresponde a cada fuente calculado dividiendo *TAC* entre el número de *N* fuentes que tienen células en el buffer, y normalizado con un parámetro de control *C* que está comprendido entre 1 y 2. Es decir, *UM= C * TAC/N* y representa la ocupación media del buffer para cada fuente cuando *C=1*.

Aunque *per-VC accounting* aporta un mejor índice de justicia a las conexiones, sólo se mantiene la asignación justa de buffer en el momento de tirar las células. Por esto es necesario conseguir la justicia en el reparto del ancho de banda entre todos los VC mediante una asignación justa del buffer en el tiempo de la transmisión. Surge así la posibilidad de distribuir el buffer mediante técnicas *per-VC queueing* cuyo comportamiento podemos ver en la *Figura 5.4*. En este caso, cada VC dispone de su propia cola en el buffer del conmutador, así como una indicación del tamaño medio de buffer ocupado por cada uno de los VC. Como puede observarse en la *Figura 5.4* se han considerado tres conexiones VC, cada una con su propia situación de ocupación de colas. Contamos también, como en el caso general de EPD, con una *CMB*, con un *Umbral general* del buffer, y con un valor de *UM* calculado como en el caso de *per-VC accounting UM= C * TAC/N*. La diferencia la tenemos en que las células de los VC son servidas a la salida del buffer siguiendo un planificador Round Robin, como puede observarse en la *Figura 5.4*, que garantiza que en cada ciclo se sirve una célula de cada uno de los VC. Esta es la técnica que va a aportar justicia a la forma de atender las fuentes. En [13,14] se realizan diversas simulaciones con el parámetro de control *C* demostrándose que a medida que se incrementa su valor, crece también el índice de justicia. Sin embargo, a medida que crece *C* también se degrada el throughput por elevarse las probabilidades de rebosamiento en el buffer del conmutador. No obstante, la investigación no demuestra la justificación ni relación directa de este parámetro con los efectos que produce.

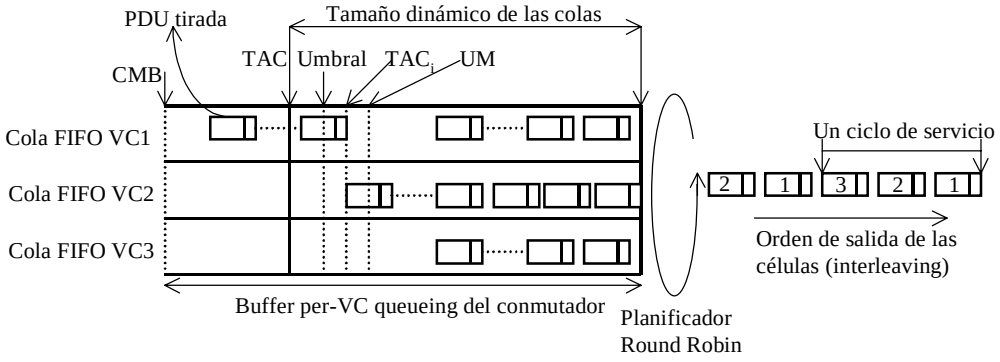

Figura 5.4. *Esquema EPD con per-VC queueing y Round-Robin*

El algoritmo que implementa el comportamiento descrito por la *Figura 5.4* es presentado en la *Figura 5.5* y, como puede analizarse, cuando se alcanza el tamaño del umbral de EPD, ningún VC deberá exceder su porción de buffer correspondiente, garantizando el grado de justicia que se busca. Además, en cada ciclo de emisión de células a la salida se va a servir una sola célula de cada uno de los VC. Así, se garantiza la justicia final de cada VC sin afectar el tamaño de cada una de las colas de los VC. Como veremos más adelante, este mecanismo aporta justicia, sin embargo, produce otro fenómeno indeseable que es el del *interleaving* ya que a la salida se mezclan los flujos pertenecientes a todos los VC. Nuestra propuesta de EPD elimina este problema además de aportar un mecanismo de retransmisión que no está soportado en ninguna de las propuestas de la literatura.

```
N=Número_de_VCs_en_el_buffer
C=Parámetro_de_Control_ (1<= C <=2)
TAC_i=Tamaño_actual_de_la_cola_de_cada_ VC_i
TAC=Suma_de_todos_los_tamaños_ TAC_i
UM=Umbral_medio_de_cada_VC
CMB=Capacidad_máxima_del_buffer_del_conmutador
Mientras una célula de un VC_i está llegando al conmutador;
    Si la célula es la primera célula de una PDU
        Calcular UM=C * TAC/N
        Si TAC >= Umbral y TAC_i >= UM
            tirar la célula
        En otro caso
            aceptar la célula en la cola del VC_i
             TAC_i = TAC_i + 1
        FSi
    En otro caso
        Si ya se ha tirado alguna célula de este paquete
            tirar la célula
        En otro caso
            Si TAC >= CMB
                tirar la célula
            En otro caso
                aceptar la célula en cola del VC_i
                TAC_i = TAC_i + 1
            Fsi
        Fsi
    Fsi
FMientras
```

Figura 5.5. *Algoritmo EPD per-VC queueing*

Hemos de destacar que también han sido propuestas en [23] variaciones de EPD en las que se establecen varios umbrales en el buffer con la intención de mejorar aún más el rendimiento. Una de estas variantes del algoritmo PPD es ESPD que va a ser descrito a continuación.

5.4. ESPD (EARLY SELECTIVE PACKET DISCARD)

Cuando se produce congestión, EPD comienza a descartar los paquetes entrantes independientemente de la sesión a la que pertenecen. EPD no se fija por tanto en la sesión que tiene mayor actividad o que consume más recursos. Sin embargo, ESPD intenta tirar sólo los paquetes de las fuentes con mayor actividad.

La referencia [7] presenta ESPD (Early Selective Packet Discard) demostrando que consigue mejorar los índices de justicia y throughput aportados por EPD, todo ello a costa de un mínimo incremento de la complejidad de implementación. La congestión continuada es el resultado de la sincronización en la expansión y reducción de la ventana de TCP. Para evitar este problema ESPD propone una estrategia para que las sesiones tengan turnos en el acceso a la capacidad de la red, tirando las células de forma selectiva en lugar de hacerlo aleatoriamente. Aunque EPD tira las células selectivamente, cuando se trata de paquetes el mecanismo para desecharlos es aleatorio, por lo que esto puede acabar afectando a pérdidas en múltiples conexiones. Estas pérdidas activan los timeout en esas sesiones TCP que deben resincronizarse, ajustar sus ventanas y acabar sobrecargando la red. Para evitar los problemas que EPD no resuelve se propone ESPD como esquema para desechar paquetes.

ESPD tiene un *reloj* de control de paquetes descartados y además incluye tres umbrales: *umbral_1_de_buffer*, *umbral_2_de_buffer* y *umbral_de_lista-descartes*.

El primer umbral del buffer se usa para capturar los VPI/VCI en la *lista-descartes* que tiene la misma función del PPD. El segundo umbral del buffer se usa para liberar los VPI/VCI de la *lista-descartes*. La función del umbral de la *lista-descartess* se usa para poner un límite al número de VPI/VCI que se almacenan en la *lista-descartes*. Por último, el *reloj* evita las situaciones de injusticia entre las fuentes de tráfico.

La *Figura 5.6* muestra el algoritmo propuesto en [7] para implementar ESPD que, como podemos observar, incrementa sustancialmente el grado de complejidad del EPD original, aunque podemos considerar que el coste de implementación es relativamente factible para las funcionalidades que aporta en la garantía de justicia y mantenimiento del buen comportamiento de la red. Los simulaciones de ESPD demuestran un mejor comportamiento que RCD, PPD y EPD. Además es capaz de encontrar un adecuado punto de equilibrio entre el índice de justicia y throughput conseguidos, de modo que no se sacrifica la justicia por incrementar el throughput.

Debemos destacar también la existencia de otras investigaciones en el contexto de EPD, como [15], donde se estudia la peor situación en que se puede encontrar EPD para conseguir mantener la integridad de los paquetes mientras se producen elevadas sobrecargas en la red. Estos trabajos demuestran que, para mantener el 100% del goodput en la red durante las sobrecargas bajo las peores condiciones posibles, se requerirá del uso de un buffer con suficiente capacidad de almacenamiento para un paquete de longitud máxima provenientes desde cada uno de los VC. No obstante, se comprueba también que puede alcanzarse el 100% del googput con buffers sustancialmente más pequeños que los que se predicen para el peor caso. Se observa también que puede conseguirse un adecuado goodput con buffers más pequeños, pero EPD experimenta malos comportamientos a medida que se reduce el tamaño del buffer e, incluso, el incremento del buffer también acaba provocando situaciones de degradación del goodput. Queda por tanto un amplio campo de investigación para determinar el punto de equilibrio en que conseguir, con el mínimo tamaño de buffer, el máximo goodput. La experiencia demuestra que las redes ATM experimentan sobrecargas por lo que es importante estudiar el funcionamiento de la red en estos periodos.

Si se observa el número de células en una cola gestionada por EPD como una función del tiempo, podrá observarse un comportamiento cíclico en el cual el número de células en la cola se incrementa por encima del umbral, entonces, como varios VC permanecen inactivos (porque han acabado sus paquetes o porque se les han tirado paquetes), el número de células detenidas incrementan y descienden dinámicamente. Cuando caen por debajo del umbral y llegan nuevos paquetes a la cola, el nivel del buffer se detiene cayendo y comenzando para elevarse de nuevo. Así, [15] presenta la siguiente expresión que permite calcular el goodput, donde p_{ok} denota el número de paquetes que pueden completarse durante un ciclo; T es la longitud del periodo de tiempo y l es el número de células en cada paquete,

$$goodput = \frac{p_{ok}}{T/l}$$

En este caso, el goodput es el ratio del número de paquetes que se completan en un ciclo para el máximo número de paquetes que podrían completarse en ese ciclo, durante el cual el enlace es usado por paquetes que son transmitidos completos. Así, en un periodo de tiempo de longitud T (donde la unidad de tiempo es el tiempo requerido para enviar una única célula), podemos enviar T/l paquetes por el enlace.

```
Mientras una célula está llegando a un conmutador;
Si el reloj_de_control ha expirado
        Borrar todas las entradas de la lista-descartes
FSi
Si el VPI/VCI de la célula está en la lista- descartes
        Si la célula es una célula EOM
                Si longitud_cola < tamaño_buffer
                        insertar la célula en el buffer
                En otro caso
                        tirar la célula
                FSi
                Si longitud_cola < umbral2_buffer
                        borrar el VPI/VCI de la lista- descartes
                        desactivar el reloj_de_control si está activo
                FSi
        En otro caso
                tirar la célula
        FSi
En otro caso
        Si longitud_cola <= umbral1_buffer
                insertar la célula en el buffer
        En otro caso Si ((# de entradas en la lista- descartes < umbral_lista- descartes
                                y célula BOM) o el buffer está lleno))
                        tirar la célula
                        capturar el VPI/VCI en la lista- descartes
                        Si el primer VPI/VCI está en la lista- descartes
                                activar el reloj_de_control
                        FSi
                        FSi
        En otro caso
                insertar la célula en el buffer
        Fsi
Fsi
FMientras
```

Figura 5.6. *Algoritmo de esquema ESPD*

5.5. RED (RANDOM EARLY DETECTION)

Para evitar los problemas de sincronización del protocolo TCP se propuso RED [8] (Random Early Detection gateways) que se encarga de tirar o marcar cada paquete que llega cuando se tiene una cierta probabilidad de ser descartado, condición que es detectada cuando la longitud media de la cola excede un umbral preestablecido. RED intenta mantener el tamaño de las colas tan bajo como sea posible, mientras se permiten ráfagas ocasionales. Se ha demostrado que RED mantiene un elevado grado de throughput mientras se consigue minimizar el retardo. La idea es monitorizar la longitud de las colas y evitar que los paquetes sean tirados cuando en la red se producen cambios. Además, garantiza que las conexiones pueden tener un alto grado de compartición del ancho de banda como se comentó en el Capítulo 4. Partiendo del trabajo original [8], en el artículo [9] se presenta una adaptación de RED a la CoS UBR, adaptando el algoritmo RED a las redes ATM.

Es conocido que las redes ofrecen mecanismos de realimentación que aportan a las aplicaciones diversas técnicas para poder conocer el estado de la red y poder monitorizar los envíos de sus datos. En las redes de conmutación de paquetes como ATM puede hablarse de dos tipos de realimentación. La explícita permite el uso de campos especiales en las células como es el caso de ABR con sus células RM para indicar congestión. Por otro lado, la realimentación implícita se basa en respuestas de la red a las variaciones en el comportamiento de las fuentes. Esta última puede ser deducida gracias a las variaciones en los retardos o por las pérdidas de los paquetes. Pues bien, RED emplea notificación de congestión implícita a través de los paquetes tirados. En lugar de esperar a que las colas se llenen y comenzar a desechar los paquetes que lleguen a partir de entonces, RED decide descartar los paquetes que llegan con una probabilidad elevada cada vez que la longitud media de las colas excede de un determinado umbral. Para cada paquete que llega al conmutador se estima el tamaño medio de la cola mediante un filtro paso bajo:

$$Long\_Media=(1-w_q)*Long\_Media+ w_q*Tamaño\_cola;$$

donde w_q es una constante que satisface $0 <= w_q <= 1$.

El algoritmo de RED es resumido en la *Figura 5.7*, donde *contador* es el número de paquetes encolados tan largos como *Long_Media* quedando entre los dos *umbrales*. El *contador* es puesto a cero en cada pérdida. El valor Max_p es la máxima probabilidad de desechado de paquetes, probabilidad que es también función del tamaño de los paquetes.

$P_a = P_b / (1 - contador * P_b)$
$P_a = Max_p * (Long\_Media - Umbral\_Mínimo) * (Umbral\_Máximo - Umbral\_Mínimo)$
$P_b = P_b * Tamaño\_Paquete / Tamaño\_Máximo\_Paquete$
Si *Long_Media <= Umbral_Mínimo*
    Aceptar el paquete
FSi
Si *Umbral_Mínimo < Long_Media < Umbral_Máximo*
    Calcular probabilidad P_a
    Tirar los paquetes que llegan con probabilidad P_a
FSi
Si *Umbral_Máximo <= Long_Media*
    Tirar los paquetes que llegan
FSi

Figura 5.7. *Algoritmo RED*

Tomando como base el algoritmo original RED, que se ha demostrado que no escala del todo bien en redes de alta velocidad, en [9] se han propuesto dos variantes C-RED y P-RED para conseguir esta escalabilidad en redes ATM, consiguiéndose reducir la complejidad de implementación del algoritmo original y obteniendo un mayor grado de justicia.

5.6. VC MERGE

Con la intención de reducir los cuellos de botella provocados en los routers, (inundados por el creciente tráfico IP), que constituyen las redes de alta velocidad, se han propuesto múltiples variantes de integración de la capa 3 de routing con la capa 2 de conmutación. La mayoría de ellas se apoyan en el establecimiento de etiquetas en la capa 2 que permitan mapearse en las tablas de routing de capa 3, dando lugar así a la evolución del clásico mecanismo de routing de capa 3 hacia la conmutación de capa 2. Con este planteamiento parece lógico que ATM sea la tecnología de capa 2 de conmutación y, de este modo, se han propuesto varias posibilidades para realizar el mapeo de la información de las rutas IP a las etiquetas ATM.

La técnica de mapeo más simple es que cada pareja inicio-destino se mapeen en un único valor de VC en el conmutador ATM. Este método es conocido como *non-VC merging* [17,18] y permite a lo receptores reensamblar células de una forma muy sencilla en los respectivos paquetes, ya que el propio valor de VC puede ser usado para diferenciar los emisores. El problema aparece cuando se necesita que esta técnica escale adecuadamente con el crecimiento de los emisores y receptores. Es decir, si tenemos *n* emisores y receptores, cada conmutador deberá soportar $O(n^2)$ etiquetas de VC para poder soportar una conectividad completamente mallada entre todos los emisores y receptores. Esto hace que las tablas de rutas VC crezcan sustancialmente a medida que crecen los nodos que se incorporan a la red (en el caso de 1.000 nodos será necesario que las tablas soporten 1 millón de entradas).

El segundo método propuesto es conocido como *VP merging* y consiste en etiquetar los VP de forma que las células etiquetadas con VP con el mismo destino se mapean al mismo valor de VP saliente en los conmutadores. Esto permite una sustancial reducción de VP ya que, para cada VP, se emplea el valor de VC como identificador del emisor con la intención de que el receptor sea capaz de reconstruir los paquetes, incluso, aunque las células de diferentes paquetes acaben experimentando el problema de interleaving. En este caso, para cada destino, un conmutador debe localizar $O(p)$ etiquetas de VP, donde *p* representa el número de puertos[5] de cada conmutador. En cualquier caso, también se depende del tamaño de la red, ya que si tenemos *n* destinos, cada conmutador deberá gestionar $O(np)$ etiquetas de VP que, desde luego, es una considerable mejora respecto *non-VC merging*. Sin embargo, aunque el espacio de etiquetas en las tablas es

[5] Un número de puertos habitual en entornos locales es 16.

razonable, el mayor problema de *VP merging* es que el espacio de VP utilizables es muy pequeño 2^{12} (4.096) en los interfaces NNI[6].

El tercer método se encarga de mapear las etiquetas de VC entrantes en los conmutadores, dirigidas a un mismo destino por la misma etiqueta de VC saliente. Esta solución es tan escalable como *VP merging O(np)* y además, no se encuentra con el problema del espacio de valores que pueden tomar los VC en la red, ya que, tanto en UNI como NNI el espacio de valores 2^{16} (65.535) es más que suficiente. Con *VC merging* las células que tienen el mismo destino no son distinguibles ni diferenciables en los puertos de salida de los conmutadores. Así, las células que pertenecen a paquetes distintos, pero que van dirigidos al mismo destino no podrán entremezclarse con otras y además, el receptor no tendrá la necesidad de realizar labores de reensamblado. Precisamente, esta característica es la que queremos aprovechar en nuestro protocolo para garantizar que los paquetes que van a salir por el mismo puerto no puedan entremezclarse con otros, por compartir el mismo valor de VC. En realidad, en nuestro caso realizamos una reinterpretación de la técnica *VC merge* aplicada al buffer, ya que nuestro objetivo es evitar que a la salida del buffer se produzcan mezclas de diferentes conexiones. Es decir, cuando un paquete sale del buffer garantizamos que va a ser tratado por completo antes de atender ningún otro. De este modo, las PDU son transferidas a su correspondiente puerto de salida sin ninguna interferencia de entrelazado con células de otras conexiones.

Como podemos comprobar, la técnica *VC merge* parece las más interesante de las tres, aunque es necesario partir de la base que requiere más espacio de buffer que *non-VC merge*, además los conmutadores tienen requerimientos hardware para soportar el reensamblado y evitar el *interleaving*. Así en [12,13] se proponen OM (Output Modules) encargados de realizar la traslación de VCI a las salidas de los conmutadores. A cada célula ATM que llega se le añaden dos campos que contienen un número de puerto de entrada y un número de puerto de salida. Partiendo del número de puerto de salida, los conmutadores reenvían cada célula al correspondiente puerto de salida de los OM. Cuando no se aplica *VC merging*, los OM se comportan como simples buffers de salida. Para desempeñar esta labor los OM disponen de RB (Reassembly Buffers) que se encargan de relacionar cada VC de entrada con un puerto de entrada, garantizando que las células de un paquete no se mezclan con las células de otros paquetes que comparten el mismo VC[7]. Durante la transferencia de un paquete hasta el buffer de salida, el VCI entrante es convertido al VCI saliente y, para ahorrar traslaciones de VCI, lo que se hace es que VCI entrantes diferentes se mezclan (merge) asignándose el mismo VCI saliente durante la traslación y siempre que las células vayan dirigidas al mismo destino. Puede optarse por una mezcla de VCI total o parcial y elegir una u otra depende de las características de las fuentes de tráfico. Por ejemplo, en el caso de tener que atender conexiones con diferentes necesidades de CoS se puede implementar merging parcial de forma que los VCI entrantes con mismo destino y misma necesidad de QoS se mapeen por el mismo VCI de salida, pero se separen de éste aquellos flujos que no requieren esa QoS como podría ser el tráfico best-effort.

La técnica *VC merge* permite que varias conexiones sean mapeadas con la misma etiqueta de VC, aportando un mecanismo escalable para soportar mucho miles de conexiones. Para escalar adecuadamente, la técnica *VC merge* requiere la presencia de buffers de reensambleado [17,18] para que las células pertenecientes a los diversos paquetes, y preparadas para el mismo destino[8], no se mezclen con otras. Así, es importante eliminar el problema de *interleaving*, aunque para ello sea necesaria la presencia de buffers adicionales que soporten la característica *VC merge*.

Las investigaciones realizadas [17] demuestran que *VC merge* genera un overhead mínimo si lo comparamos con *non-VC merging* en términos de buffers adicionales. En realidad, el overhead desciende a medida que el tráfico se incrementa o a medida que se generan más ráfagas. Pero también se ha demostrado que el retardo adicional en que se incurre por aplicar este mecanismo es mínimo para la mayoría de aplicaciones.

Otras interesantes investigaciones [19-22] han realizado propuestas en la línea de la técnica de merging con variantes hacia el problema del multicasting como el *compound VC* introducido en [19,22] para integrar el IP multicast sobre ATM multicast nativo. En este caso los grupos multicast de IP son asociados a grupos de VC (compound VC) de forma que, con una sóla entrada en la tabla de conmutación, se puede conmutar el tráfico de un grupo multicast completo gracias al uso de máscaras. Se emplea multiplexación ID per PDU como en [21], pero aportando mayor escalabilidad en cuanto al número de grupos multicast que se pueden soportar. Se evita el *interleaving* de las células mediante la asignación dinámica de identificadores de cada PDU.

---

[6] La longitud del campo VPI de 12 bits en las células ATM limita sustancialmente la capacidad de direccionamiento de VP merging.

[7] Este mecanismo se conoce con el nombre de store-and-forward de paquetes.

[8] En este contexto la palabra destino se refiere a la red destino (prefijo CIDR), pero puede también entenderse como mismo nodo destino, mismo puerto de destino, mismo usuario destino, misma CoS destino, misma QoS destino, etc.

5.7. PROPUESTA EPDR (EARLY PDU DISCARD AND RELAY) PARA TAP

Como ya hemos comentado, la arquitectura TAP incorpora un esquema de control del buffer basado en EPD. Nuestro algoritmo es novedoso respecto al esquema EPD original y sus variantes existentes porque:

- Incluye un mecanismo de retransmisión cuando se detectan PDU que van a provocar el rebosamiento del buffer. Por tanto, cuando se produce la situación de buffer lleno, la PDU que es tirada será automáticamente solicitada al conmutador previo para ser retransmitida. De este modo, se optimiza el goodput de una forma muy superior a cualquier otro esquema de la literatura que suponen que la retransmisión será realizada extremo-extremo. Las retransmisiones sólo se solicitan en el caso que la fuente tenga suficiente tiempo de inactividad para atenderlas sin afectar a la transmisión en curso. Así, nuestra propuesta permite ahorrar, en el peor de los casos, el tiempo *RTT extremo-extremo* y además, el proceso de solicitud de retransmisión de los protocolos de las capas superiores como TCP. Y lo que es más importante, la implosión en los emisores de tráfico es solventada por la delegación realizada en los conmutadores AcTMs que soportan la arquitectura TAP.
- El problema de la fragmentación de las PDU es solventado empleando el campo EOM de las EAAL-5 que es otra de las propuestas del protocolo que soporta TAP. De este modo se reduce al máximo la posibilidad de desechar PDU, aunque la fragmentación no es completamente eliminada como en el resto de variantes de EPD.
- El problema de la justicia no es un objetivo del esquema de control del buffer ya que esta labor es desempeñada por el algoritmo QPWFQ comentado en el capítulo anterior. Por tanto, la justicia está garantizada, pero también la caracterización del tráfico que requiere que ciertas fuentes de tráfico tengan prioridad sobre otras en función de los pesos asignados a cada una de las colas de entrada de la arquitectura.
- El buffer es controlado por un agente programable que permite la coordinación con otros agentes del sistema para ajustar los umbrales más convenientemente en función de las situaciones del estado de cada conmutador. Este agente permite al operador de la red dimensionar el buffer y elegir el esquema a aplicar. En estos momentos puede elegirse entre PPD y EPDR.
- El problema del interleaving es evitado por una variante de *per-VC queueing* que nos permite procesar las PDU existentes en el buffer de forma separada sin provocar la mezcla de células a la salida. De todos modos, este problema también es amortiguado por el esquema QPWFQ que se encarga de enviar al buffer de forma continua todas las células de la misma PDU. Además, QPWFQ ya atiende el tráfico aplicando técnicas *per-VPI/VCI*.
- El coste algorítmico es similar a los esquemas que hemos tenido la ocasión de revisar en apartados previos de este capítulo. En cuanto a requerimientos hardware tampoco necesitamos recursos extraordinarios.

Todos estos aspectos serán discutidos datalladamente en capítulos siguientes, pero queremos avanzar tanto el algoritmo, como una representación gráfica de la sección de la arquitectura que se encarga de esta labor en los conmutadores AcTMs.

La *Figura 5.8* ilustra el algoritmo que implementa el esquema EPDR de control del buffer basado en [7] en el que hemos introducido la función *Retransmitir(indexPDU)* que describiremos también detalladamente en el *Capítulo 10*. El valor del umbral es determinado por el agente programable CCA en función de la determinación del operador de la red y del estado de la propia red. Como podemos comprobar en el algoritmo, en cuanto la longitud actual de la cola iguala el valor del umbral, el conmutador se dispondrá a tirar todas las células de la PDU que ha provocado la situación. Pero en lugar de empezar a descartar células en cuanto se iguala el umbral, se espera a que llegue la primera célula de una PDU (BOM de AAL-3). Es decir, cuando llega la primera célula de una PDU y ya se ha igualado el umbral, se tira esa célula y se marca el VPI/VCI de la célula en la *lista-descartes* para, a continuación, tirar todas las células de esa misma PDU hasta llegar a su última célula (EOM). Este es el modo en que se garantiza que no se produzca la fragmentación de la PDU, impidiendo que no se envíe ninguna célula de una PDU que ha provocado una situación de congestión. Una vez ha llegado la última célula de la PDU congestionada, y puede accederse al campo PDUid, se solicita la retransmisión de la PDU a través del agente RCA. Una vez solicitada la retransmisión de la PDU se procede a eliminar el valor VPI/VCI de la *lista-descartes* para que la siguiente PDU de esa misa conexión pueda ser aceptada en el buffer si ya ha pasado la situación de congestión.

Como puede comprobarse también, la célula EOM de la PDU congestionada se introduce en el buffer si tiene espacio para ello, ya que esta célula es la que usa EAAL-5 (como el estándar AAL-5) para delimitar

unas PDU de otras. De este modo se sabe cuándo acaba una PDU congestionada. Destacamos también que, al igual que en las propuestas originales de EPD, el algoritmo EPDR no solventa la fragmentación cuando una vez insertadas las células iniciales de una célula se llena el buffer. La mejor forma de evitar que esto se produzca es la elección adecuada del valor de umbral, que debería ser menor en unas tres o cuatro PDU que el límite del buffer. No obstante, aunque puedan seguir adelante las células iniciales de la PDU, las restantes células hasta llegar a la célula EOM no serán transmitidas para no seguir enviando células de una PDU que se sabe ya corrompida.

```
Mientras una célula llega al buffer
Si el VPI/VCI de la célula pertenece a la lista-descartes
    Si la célula es una célula EOM
        Si Longitud_de_Cola < Tamaño_Buffer
            insertar la célula en el buffer
        En otro caso
            tirar la célula
        FSi
        Retransmitir(indexPDU)
        borrar el VPI/VCI de la lista-descartes
        FSi
    En otro caso
        tirar la célula
    FSi
En otro caso
    Si Longitud_de_Cola < umbral
        insertar la célula en el buffer
    En otro caso Si (primera célula de PDU  o (el buffer está  lleno))
                tirar la célula
                capturar el VPI/VCI en la lista-descartes
            FSi
    En otro caso
        insertar la célula en el buffer
    FSi
FSi
FMientras
```

Figura 5.8. *Esquema de planificación del algoritmo EPDR*

La *Figura 5.9* muestra una representación gráfica del buffer en el que podemos observar todos los elementos que intervienen. Podemos observar el punto de llegada del tráfico de entrada, así como los límites de los valores de la *Longitud Actual de Cola* (*LAC*), el valor *Umbral* (*U*) y el valor de *Tamaño Máximo del Buffer* (*TMB*). También puede comprobarse la comunicación existente entre el buffer y los agentes WFQA, CCA, RCA y DPA, así como la conexión del buffer con la memoria DMTE en la que se realizan las copias de las PDU que han llegado completas al buffer.

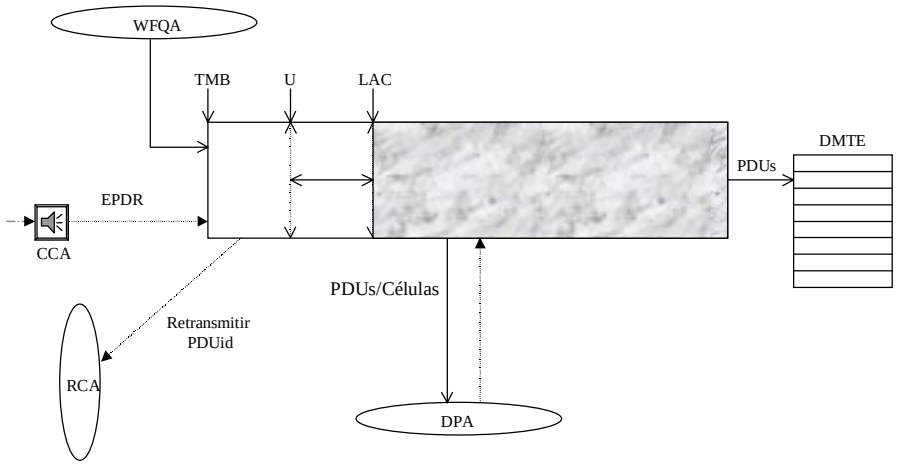

Figura 5.9. *Esquema gráfico del buffer*

Por último destacamos que en nuestro caso el VC-merging se basa en *store-and-forward* de paquetes en el buffer, de forma que se almacenan las células de cada paquete que llegan al buffer hasta que llega la última célula de cada PDU. Cuando llega la última célula de una PDU se transfieren de una forma atómica [9] al correspondiente puerto de salida del conmutador. Desde el punto de vista de la implementación, las operaciones atómicas nos suponen no mucho más que la sincronización de dos agentes software y el movimiento de un puntero en el buffer.

5.8. CONCLUSIONES

Resumiendo, el hecho de que los protocolos extremo-extremo envíen información en paquetes conteniendo múltiples células ATM provoca que el impacto de periodos de sobrecarga empeoren aún más el comportamiento de la red porque la pérdida de una sola célula puede conducir a la pérdida y retransmisión entre los extremos de la PDU completa de la capa de transporte. Esto hace que durante los periodos de sobrecarga, las redes ATM puedan experimentar congestión y colapso. Mediante los esquemas de control del buffer pueden garantizarse el goodput y la justicia entre las diversas fuentes de tráfico. En nuestro caso incorporamos un algoritmo que solventa los problemas de fragmentación, implosion e interleaving.

REFERENCIAS

[1] M., Hluchyj, "Closed-Loop Rate-based Traffic Management," *ATM Forum Contribution 94-0438R2*, (1993).

[2] R. Jain, S. Kalyanaraman, and Viswanathan, "Simulation Results: The EPRCA+ scheme," *ATM Forum Contribution 94-0988*, (1994).

[3] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, and H. Miyahara, "Rate-based Congestion Control for ATM Networks," *Computer Communications Review*, pp. 60-72, (1995).

[4] ____ ATM Forum, "Traffic Management Specification Version 4.0," *ftp://ftp.atmforum.com/pub/approved-specs/aftm-0056.000.ps*, (1996).

[5] G. Armitage and K. Adams, "Packet Reassembly during Cell Loss," *IEEE Networks*, vol. 7, no 5, pp. 26-34, Sept. (1993).

[6] Allyn Romanov and R. Oskouy, "A performance enhancement for packetized ABR and VBR+data," *AF-TM 940295*, (1994).

[7] Kangsik Cheon and Shivendra S. Panwar, "Early Selective Packet Discard for Alternating Resource Access of TCP over ATM-UBR," *IEEE LCN'97*, pp. 306-316, (1997).

[8] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, pp. 397-413, (1993).

[9] Omar Elluomi and Hossam Afifi, "RED Algorithm in ATM Networks," *IEEE SIGCOMM'2000*, pp. 312-319, (2000).

[10] José Luis González-Sánchez and Jordi Domingo-Pascual "RAP: Protocol for Reliable Transfers in ATM Networks with Active Switches," *International Conference on Communications in Computing (CIC'2000)*, pp. 141-148, (2000).

[11] Allyn Romanov and Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal on Selected Areas in Communications*, pp. 633-641, (1995).

[12] R. Goyal, G. Jain, S. Fahmy, and S. Kim, "Performance of TCP over UBR+," *AF-TM 96-1269*, (1996).

[13] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, C. Ikeda, and H. Suzuki, "Performance of TCP over UBR service in ATM networks with Per-VC Early Packet Discard schemes," *Proceedings IEEE IC3N*, pp. 350-357, (1996).

[14] Yuang Wu, Kai-Yeung Siu and Wenge Ren, "Improved Virtual Queueing and Dynamic EPD Techniques for TCP over ATM," *IEEE International Conference on Network Protocols*, pp. 212-219, (1997).

[15] Maurizio Casoni and Jonathan S. Turner, "On the Performance of Early Packet Discard," *IEEE Journal on Selected Areas in Communications*, Vol. 15, no 5, pp. 892-902, (Jun. 1997).

[16] Maurizio Casoni "Early Packet Discard with diverse management policies for EOM cells," *IEEE Proceedings International Workshop on Broadband Switching Systems*, pp. 33-37, (1997).

[17] I. Widjaja, and I. I. Elwalid, "Performance issues in VC.merge capable switches for IP over ATM networks," *IEEE Proceedings INFOCOM'98*, pp. 372-380, (1998).

[18] I. Widjaja, and A. I. Elwalid, "Performance issues in VC.merge capable switches for multiprotocol label switching," *IEEE Journal on Selected Areas in Communications*, pp.1178-1189, (1999).

[9] La atomicidad la implementamos como el envío independiente de cada paquete a la salida en una sola operación.

[19] Josep Mangues-Bafalluy and Jordi Domingo-Pascual, “Performance Issues of ATM Multicasting Based on Per-PDU ID assignment,” *Proceedings IEEE Int’l Conference Communications (ICC’00)*, (2000).

[20] M. Baldi, D. Bergamasco, S. Gai, and D. Malagrinò, “A Comparison of ATM Stream Merging Techniques,” *Proceedings of IFIP High Performance Networking*, pp. 212-227, (1998)

[21] J. Calvignac, P. Droz, C. Baso, and D. Dykeman, “Dynamic IDentifier Assignment (DIDA) for merged ATM connections,” *ATM Forum 97-0316* (1997).

[22] Josep Mangues-Bafalluy and Jordi Domingo-Pascual, “Compound VC Mechanism for Native Multicast in ATM Networks,” *Proceedings of the ICATM’99*, pp.115-124, (1999).

[23] Jonathan S. Turner, “Maintaining High throughput during overload in ATM switched,” *Proceedings IEEE INFOCOM’96*, pp. 287-295, (1996).

[24] RR. Jain, “Congestion control and traffic management in ATN networks:Recent advances and a survey,” *Comput. Networks ISDN Syst.*, (1996).

CAPÍTULO 6

AGENTES SOFTWARE Y REDES ACTIVAS

6.1. INTRODUCCIÓN

En los últimos años se ha producido una dinámica actividad investigadora en el campo de los agentes aplicados a muy diversas actividades. El concepto de agente puede ser estudiado desde muy distintos puntos de vista por lo que no existe un consenso claro en su definición, aunque sí en la idea intuitiva que todos tenemos de este atractivo campo de investigación. Uno de los ámbitos de acción de los agentes es precisamente el de los sistemas de telecomunicación, porque los agentes pueden dotar a las redes de características de las que carecen en la actualidad. Informalmente, puede decirse que un agente es una entidad con objetivos, que alcanza ejecutando una serie de acciones, mediante un dominio de conocimiento y situado en un entorno concreto. De este modo, podríamos acordar que un sistema multiagente es un grupo de agentes situados en un entorno donde puede desarrollarse algún tipo de comunicación entre ellos, bien directamente, o bien a través del entorno como resultado de alguna de las acciones llevadas a cabo por los propios agentes. No obstante, estas dos definiciones pueden resultar demasiado laxas dependiendo del contexto y, por esto, deseamos aclarar una serie de conceptos importantes, así como presentar las clasificaciones más aceptadas de agentes para concluir presentando de forma resumida el sistema multiagente TAP que proponemos para dar soporte a nuestra arquitectura.

Por otro lado nos encontramos con las redes activas basadas en la posibilidad de equipar a los nodos que las constituyen con características que los convierten en elementos activos. Como en el caso de los agentes, falta también el consenso para decidir lo que es una red activa. No obstante, disponemos de criterios para determinar algunas características que nos servirán para reafirmar nuestros objetivos.

Aunque existen muchos puntos de conexión entre el conceptos de agentes aplicados a las redes de comunicaciones y el de redes activas, existe cierta controversia entre ambos campos de aplicación. No obstante, veremos cómo en nuestro caso, con la implementación del sistema multiagente TAP soportado en los conmutadores ATM activos (AcTMs) que hemos diseñado, conseguiremos dotar a la tecnología ATM de esas características activas que le aportan la interacción, autonomía y adaptación de los agentes software que proponemos. Centraremos por tanto nuestro campo de acción en el concepto más general de los agentes software aplicados a las redes, dando así lugar a lo que llamamos redes programables.

El objetivo de este capítulo consiste en la revisión de estos dos campos de investigación, sin entrar en cuestiones encontradas, ni en elecciones concretas en cuanto a los estándares a usar. Así, nos centraremos en el concepto de agentes software, aplicados a las redes de comunicación en general y a ATM en particular, para conseguir que éstas se conviertan en activas. Fundamentamos de este modo el adjetivo activo del nombre de la arquitectura TAP. Para ello este capítulo presenta en primer lugar los fundamentos del concepto de agente, para pasar después a estudiar los sistemas multiagente como sociedades de agentes. El cuarto apartado establece las características de comunicación entre agentes, para pasar después a describir algunas cuestiones relativas a las arquitecturas. Una vez revisadas las propuestas estándares en la materia, se pasa a comentar los beneficios del uso de agentes en el ámbito de las redes. Seguidamente, se realiza una revisión de diversos agentes diseñados para los sistemas de comunicaciones. Estudiados estos aspectos se comentan los puntos de vista de las redes activas; se analiza y caracteriza la arquitectura del Sistema multiagente TAP que proponemos, y el capítulo termina con un apartado de conclusiones.

6.2. CONCEPTOS FUNDAMENTALES SOBRE AGENTES

El paradigma de los agentes software tiene su origen en la década de los años 70 como una corriente de la Inteligencia Artificial Distribuida (IAD). Sin embargo, desde los primeros trabajos sobre agentes y hasta la actualidad, han aparecido varias disciplinas que se han encargado de definir y delimitar su ámbito de acción y, entre ellas, queremos destacar las siguientes:

- Inteligencia artificial, especialmente en el ámbito distribuido.
- Programación orientada a objetos y programación concurrente.
- Telecomunicaciones.
- Diseño de interfaces hombre-máquina.
- Entornos de producción industrial.

Pero el primer problema al que nos enfrentamos es precisamente a la definición del concepto de agente, ya que existen diversas acepciones para esta palabra y no disponemos de una definición única ni definitivamente aceptada, con la consiguiente controversia. En lo referente a la idea intuitiva de lo que representa el término agente parece que el consenso sea mayor, pero -no obstante- vamos a presentar algunas de las definiciones más representativas de la literatura para movernos en un terreno firme desde el punto de vista conceptual.

El Diccionario de la Lengua Española [1] realiza la siguiente definición de *agente*:

"*Agente: Del lat. agens, -entis, p. a. de agere, hacer. 1.adj. Que obra o tiene virtud de obrar. 4.m. Persona o cosa que produce un efecto. 5.Persona que obra con poder de otro...*".

El Diccionario Webster [2] define la palabra *agent* con las siguientes acepciones:

".n. ***1****. a person authorized by another to act on his behalf.* ***2****. one who or that which acts or has the power to act.* ***3****. A natural force or object producing or used for obtaining specific results.* ***4****. An active cause; an efficient cause.* ***5****. One who works for or manages an agency".*

El Agent Technology Glossary del *Object Management Group* (OMG) [3, URL1] destaca la siguiente definición popular:

"*Un agente o agente software es un programa software que hace algo, generalmente en nombre de una persona o de otro agente,*

- *posiblemente automatizando alguna tarea (por ejemplo indexando),*
- *posiblemente usando cierta inteligencia (por ejemplo planificando o negociando),*
- *posiblemente activado por algún evento que es monitorizado o ejecutado como tarea de fondo haciendo alguna tarea (por ejemplo seleccionando y filtrando información de interés),*
- *posiblemente moviéndose de un sitio a otro (para recopilar información),*
- *posiblemente comunicando o interactuando con el usuario (a través de charla o diálogo),*
- *posiblemente comunicando o interactuando con otros agentes de una forma coordinada y cooperativa,*
- *posiblemente aprendiendo y cambiando su comportamiento a lo largo del tiempo (en respuesta a cambios en el entorno),*
- *posiblemente operando con iniciativa propia y generando informes periódicos".*

Por otro lado, Franklin y Graesser indican [4]:

"*Un agente autónomo es un sistema dentro de un entorno, que siente el entorno y actúa dentro de él siguiendo su propia agenda para conocer el efecto de lo que sentirá en el futuro*".

Además, Hayes-Roth [4] destacan que:

"*Los agentes inteligentes realizan continuamente tres funciones: percepción de las condiciones dinámicas de un entorno; acción que afecta a las condiciones del entorno; y razonamiento para interpretar percepciones, resolver problemas, realizar inferencias y determinar acciones*".

Desde un punto de vista más pragmático el IBM's Intelligent Agent Strategy, Aglets [4] concluye:

"*Los agentes inteligentes son entidades software que desempeñan una serie de operaciones en nombre de un usuario o de otro programa, con algún grado de independencia o autonomía, empleando algún conocimiento o representación de los objetivos o deseos del usuario*".

A la pregunta de ¿*qué es un agente*? ya respondió Carl Hewitt: "*...para la comunidad que investiga en la computación basada en agentes esta es una cuestión tan embarazosa, como lo es para la comunidad dedicada a la inteligencia artificial el responder a ¿qué es la inteligencia?*". Sin embargo, OMG [3] parte de un planteamiento más simplista indicando que "*un agente es alguien que actúa*". Aunque también se precisa que para su uso práctico, los agentes deben poseer una serie de propiedades que van a ser comentadas a lo largo de este capítulo.

Ante la situación que acabamos de describir, el mayor inconveniente que encontramos es que estamos ante un término ampliamente usado por muchos investigadores en campos muy diversos, lo que impide poder dar una definición universalmente aceptada. Wooldridge y Jennings [5] responden a la pregunta anterior sin recurrir a definiciones magistrales y se centran en describir las características que deben tener los agentes para ser considerados como tal. Estas características serán comentadas más adelante una vez centrados ciertos aspectos que consideramos de interés.

Especialmente destacable para nosotros es el planteamiento de [6] donde se indica que "*las redes de comunicaciones futuras se conciben más automatizadas, pudiendo ser gestionadas por entidades software, tanto móviles como estáticas, coleccionando información de estado de la red y que aportan la habilidad innata para invocar cambios efectivos en los controladores de los conmutadores sin la interacción explícita de ningún operador humano.*" Este es el modelo planteado por investigadores y desarrolladores de tecnologías[1] como TINA-C (Telecommunications Information Networking Architecture-Consortium) [URL2] y Active Networks [7, URL3]. Para hacer realidad estos modelos, las plataformas de telecomunicaciones intentan abarcar el, relativamente nuevo, paradigma de los agentes software.

No es nuestro principal objetivo el de describir aquí las propuestas que en los últimos años ha realizado la Inteligencia Artificial en el ámbito de la tecnología de los agentes software, pero sí consideramos oportuno aclarar algunos de los conceptos básicos para centrar y afianzar adecuadamente nuestras investigaciones en el contexto adecuado. Nwana presenta en [8], y en otros trabajos previos, los fundamentos y los campos de aplicación de los agentes software. La referencia [6] realiza una detallada descripción de los aspectos más importantes de los agentes inteligentes desde los planteamientos teóricos, hasta sus arquitecturas y lenguajes de implementación. En [9] se actualizan y revisan las últimas arquitecturas en materia de agentes. Existen otras muchas investigaciones y publicaciones relacionadas con los aspectos de la Inteligencia Artificial y la IAD, donde se discuten muy diversas características de los agentes inteligentes. La revisión de toda esta literatura nos permite determinar que el ámbito de las telecomunicaciones ofrece un atractivo campo de aplicación a las técnicas de los agentes software, tanto en el presente como para el futuro.

Pero antes de seguir adelante cabe distinguir entre varios conceptos que tienden a confundirse por sus numerosos puntos de conexión. Trataremos por tanto de aclarar los conceptos de Agentes Software, Agentes Inteligentes, Agentes Móviles, Arquitecturas de Agentes y Sistemas Multiagentes.

Esta sección aclara los conceptos y nociones que permiten definir lo que es un agente software, así como sus aplicaciones en diversos ámbitos de los sistemas de comunicaciones, centrándonos principalmente en la tecnología ATM. El término agente software ha sido adoptado [10] "*como la frase más general para describir el concepto de una entidad software que automatiza algunas de las tareas consideradas como más mundanas o laboriosas para un agente humano*". No obstante, en no pocos casos se ha complementado el concepto anterior considerando dos variantes o ampliaciones del mismo, como son los llamados Agentes Móviles y los Agentes Inteligentes Cooperativos o Agentes Inteligentes.

Como veremos, existen importantes diferencias entre estas tres variantes que pueden ser usadas para solventar muy distintos problemas. En cualquier caso, la literatura [11] describe también el concepto general de Agentes Móviles Inteligentes como una importante vía de investigación sobre los sistemas de telecomunicación futuros.

En nuestro caso no deseamos realizar una exposición detallada de los aspectos formales y fundamentales de los agentes software sino, más bien, centrarnos en el caso concreto en que nuestras aplicaciones van a hacer uso de agentes software en la activación de gestión automatizada y el control de sistemas de telecomunicaciones aplicados a la tecnología ATM. Publicaciones como [4-6,8-15] presentan aspectos mucho mas detallados de los conceptos de agentes, sus amplias áreas de aplicación, su formalismo, etc.

[1] Tecnología puede ser considerada, en este caso, como un conjunto de métodos para lograr ciertos objetivos relacionados.

Otra de las múltiples definiciones de agentes la podemos encontrar en [10], donde se describe un agente software como un *programa ejecutado independientemente capaz de actuar autónomamente sin supervisión humana directa en tiempo de ejecución, siendo capaz de seleccionar acciones cuando se producen eventos esperados o limitadamente inesperados*. Pueden tener diversas habilidades pero, sobre todo, suelen tener la peculiaridad de poder relacionarse dentro de su propio mundo (entorno operacional o dominio de aplicación). El término agente se ha usado también en el ámbito de los sistemas distribuidos para referirse a las entidades usadas para realizar tareas muy concretas. La referencia [5] es la descripción ya clásica para determinar lo que son los agentes software, destacando sus características de habilidad social, autonomía, reactividad y adaptabilidad.

Por otro lado, [12] presenta una interesante taxonomía de los agentes en el ámbito de los sistemas de telecomunicaciones realizando su división en: agentes locales, agentes de red, agentes basados en IAD y agentes móviles.

En líneas generales, cabe destacar que los investigadores de la Inteligencia Artificial opinan que un agente ideal debería tener tres cualidades destacables: posibilidad de cooperar con otros agentes o programas, capacidad de aprendizaje y también autonomía [14]. Sin embargo, y como ya se ha destacado, el consenso no es sencillo en este aspecto y por eso se consideran para la definición de agentes las siguientes características básicas [15]:

- Situación: que permita localizar a los agentes en un entorno determinado.
- Flexibilidad: de forma que las acciones no tengan que estar prefijadas.
- Autonomía: que les permita actuar sin la obligación de requerir la intervención directa del hombre o de otro programa.

Las definiciones y características ya comentadas permiten realizar una clasificación de los agentes con la intención de determinar el ámbito de actuación y aplicación de éstos. En esta línea se han hecho también múltiples clasificaciones de agentes que dependen de los investigadores que las realizan, pero podemos citar la presentada en [URL1] en función de las propiedades que satisface cada agente, destacando que no todos los agentes deben cumplir todas estas propiedades:

- Autonomía: los agentes son proactivos[2], siendo capaces de tomar la iniciativa para conseguir sus objetivos por sí solos sin requerir iniciación del usuario, confirmación ni notificación. Es decir, pueden actuar sin intervención externa directa, tienen un cierto grado de control sobre su estado interno y sus acciones están basadas en su propio conocimiento. No sólo reaccionan al entorno, sino que también son orientados a objetivos o propósitos propios.
- Movilidad: capacidad de moverse a donde necesitan, posiblemente siguiendo un itinerario.
- Adaptabilidad: cuando el agente es capaz de adaptarse dinámicamente aprendiendo de su entorno ante la incertidumbre y los cambios. Es decir, la capacidad de responder a otros agentes y/o al entorno en un cierto grado. Incluso, formas avanzadas de adaptación permitirán al agente modificar su comportamiento basándose en su propia experiencia.
- Reacción: actuación sobre el entorno mediante un comportamiento estímulo/respuesta.
- Delegación: pueden actuar en representación de alguien o algo.
- Inteligencia: estado formalizado por el conocimiento (creencia, objetivos, planificación, etc.)
- Sociabilidad: capacidad para colaborar en comunidades para conseguir objetivos comunes.
- Racionalidad: capacidad para elegir una acción basándose en objetivos internos o en el conocimiento.
- Coordinación: con capacidad para realizar actividades compartidas con otros agentes del entorno. La actividad de coordinación se realiza a través de planes o por otros mecanismos de gestión.
- Cooperación o colaboración: capacidad para coordinarse con otros agentes para alcanzar objetivos comunes, es decir, agentes no rivales que alcanzan conjuntamente el éxito o el fracaso.
- Competitividad: es decir, el opuesto a cooperación, de modo que el éxito de un agente lleva al fracaso de otros.

[2] Suele emplearse el término proactivo en contraposición al concepto reactivo que indica que los agentes se limitan a reaccionar ante situaciones previamente determinadas. De este modo, los agentes proactivos podría decirse que son algo más que activos.

- Interacción: interoperación con humanos, otros agentes, sistemas legales y fuentes de información. En suma, la capacidad para comunicar con el entorno y con otros agentes del entorno.
- Impredecibilidad: de modo que pueden actuar sin que se pueda predecir completamente su comportamiento. Es decir, un comportamiento no determinista.
- Continuidad: como procesos de ejecución continua en el tiempo.
- Transparencia y contabilidad: es decir, serán transparentes cuando se requiera, pero ofrecerán un registro de sus actividades si se solicita.
- Carácter: inclusión de estados emocionales y de personalidad.

Por otro lado, la literatura más centrada en el ámbito de la IA suele presentar los agentes divididos en una clasificación a seis niveles:

- Agentes móviles o estáticos.
- Agentes reactivos o proactivos.
- Agentes cooperativos, autónomos, con aprendizaje o con una mezcla de estas características:
  - ✓ Agentes de interfaz: autónomos y con aprendizaje.
  - ✓ Agentes colaboradores: autónomos y cooperativos.
  - ✓ Agentes ideales: autónomos, cooperativos y con aprendizaje.

En cualquiera de los casos los agentes software son entes autónomos (ver *Figura 6.1*) con sensores para percibir (interactuar) entradas de información o eventos procedentes del entorno o de otros agentes y actuadores[3] para efectuar las operaciones de salida sobre el entorno u otros agentes. También tienen habilidades computacionales (razonamiento, búsqueda, etc.) y pueden usar sus modelos racionales y de conocimiento para mapear las entradas en salidas pudiendo maximizar sus utilidades (suele medirse el rendimiento de acuerdo a la racionalidad).

Aunque no existe una clara definición del concepto agente podemos decir que, en el ámbito de las tecnologías de la información, los agentes deben cumplir, como poco, las caracterísiticas de autonomía, interactividad y adaptabilidad, comentadas anteriormente, para ser considerados como tales.

En nuestro caso concreto, con la aplicación general de los agentes a las TI y, concretamente, a las TIC, podría hablarse de las siguientes formas de agentes, ya sean software, hardware o ambos:

- Agentes móviles son aquellos capaces de mover su código a un nodo diferente a donde residen, mientras un agente estático permanece en un solo computador o nodo. En determinadas aplicaciones la movilidad puede ofrecer atractivas ventajas, aunque supone también ciertos inconvenientes relacionados con la seguridad del sistema, con el hecho, de que al mover el código de un agente a otro nodo, se consumen recursos en la comunicación y en el nuevo nodo (CPU, memoria, etc.) en el que se va a alojar el agente.

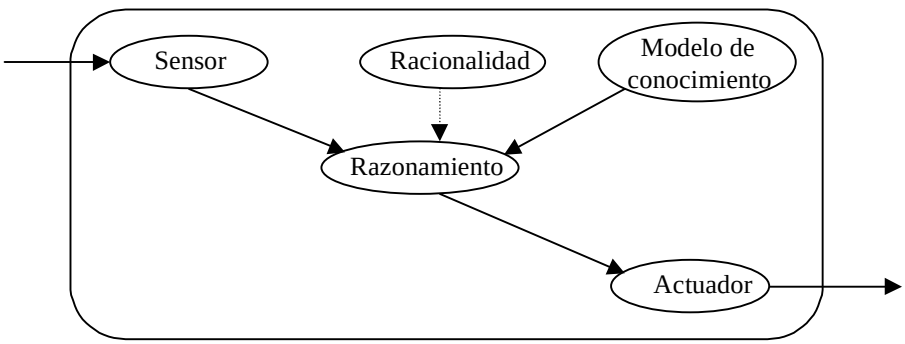

Figura 6.1. *Modelo general de agente software*

[3] Del ingles *effector*.

- Agentes software: son el tipo de agente más específico y, aunque existen también múltiples definiciones, podemos partir de la realizada por OMG [3] en la que se indica que un agente software es una entidad que puede interactuar con su entorno. Es decir, existen agentes que pueden ser implementados usando software, lo que significa que pueden ser autónomos y pueden reaccionar con otras entidades, incluyendo los humanos, máquinas y otros agentes software en varios entornos o plataformas. Debemos destacar que este planteamiento es el que nosotros usamos en nuestra propuesta de forma que nos referiremos a la arquitectura TAP como constituida por agentes software.

- Agentes autónomos: Según la definición de autonomía comentada antes, un agente será autónomo si dispone de independencia de controles externos, es decir, si opera sin intervención ni invocación externa. En realidad, la autonomía se comprende mejor si se especifica en el grado de que se dispone para realizar ciertas acciones. De este modo, suele distinguirse entre autonomía dinámica, y autonomía no determinística, que es impredecible. Un agente es dinámico si es capaz de ejercer algún grado de actividad, pero también puede ser simplemente pasivo o ser completamente proactivo[4] (pueden elegir abiertamente sus acciones).

- Agentes interactivos son los que cumplen la ya citada característica de interactividad, que también puede ser expresada en determinados grados. El caso más básico de interacción puede ser el método de invocación de los mensajes entre objetos. Y un caso más complejo puede ser la posibilidad que un agente reaccione tras observar ciertos eventos que se producen en su entorno. Esta escala de interactividad puede crecer considerablemente hasta llegar a sociedades de agentes con múltiples interacciones paralelas entre todos ellos con comportamientos competitivos, colaborativos, negociadores, etc.

- Agentes adaptativos son aquellos que, como mínimo, son capaces de reaccionar a un estímulo. Esto es, deben ser capaces de realizar una respuesta predeterminada a un evento particular o señal del entorno. Un sensor de temperatura, o un detector de humos, o un *daemon* de Unix, podrían ser uno de estos ejemplos. Es decir, además de los agentes reactivos podemos tener agentes que pueden razonar. Los agentes que razonan pueden ser también reactivos, sin embargo, se caracterizan por su capacidad para reaccionar realizando inferencias[5]. Las formas más avanzadas de adaptación incluyen la capacidad de aprendizaje y evolución basadas en técnicas de redes neuronales y algoritmos genéticos respectivamente.

- Agentes coordinativos o con la capacidad de coordinar sus acciones con otros agentes para alcanzar un propósito. Cualquiera de las actividades humanas requiere de un trabajo coordinado para conseguir objetivos generales y, por analogía, este planteamiento es también válido en el ámbito de los agentes. Un ejemplo coordinativo puede ser una cadena de montaje de vehículos, o la mayor parte de las actividades de fabricación, pero también se necesita coordinación en la mayor parte de protocolos de comunicaciones para conseguir el objetivo final de cualquier red de comunicación.

- Agentes inteligentes son los que están dotados de inteligencia. Aunque para bastantes investigadores la palabra agente es equivalente a agente inteligente, el mayor inconveniente lo encontramos en la falta de acuerdo en la definición de la palabra inteligencia en el ámbito de la IA. De hecho, el OMG no dispone de una definición normalizada del concepto de agente inteligente, por lo que existen también varios tipos o grados de inteligencia de los agentes. En cualquiera de los casos, se parte de la idea que los agentes requerirán de un conjunto básico de atributos y facultades que se indicaron antes en la capacidad de inteligencia. Un agente inteligente debería ser capaz de examinar sus creencias y deseos, formalizar sus intenciones, planificar las acciones que desea realizar basándose en ciertas suposiciones y actuar para lograr sus planes.

- Agentes envolventes[6] ofrecen una vía para legar código y aportar la posibilidad de reusar código a un proceso, o transformar un trozo de código estático en una entidad activa. Es decir, este tipo de agentes aporta a otros agentes una capa, interfaz o envoltorio para conectarse a software que no puede ser considerado como agentes. En suma, estos agentes aportan la posibilidad de que los agentes interactúen con las partes de un sistema que no son agentes.

[4] Aunque las abejas tienen principalmente un comportamiento reactivo, pueden tener también un cierto grado de proactividad cuando eligen comer o volar o picar o reproducirse, etc.

[5] Sacar una consecuencia de un hecho o de un principio. La deducción es una inferencia.

[6] Del inglés wrapper para indicar el concepto de interfaz o capa entre diferentes tipos de código.

Se han descrito muchas otras formas de agentes como agentes administradores, facilitadores, colaborativos, etc, pero las que acabamos de comentar son las más ampliamente usadas en la literatura para poder determinar cuándo se está o no ante un agente.

6.3. SISTEMAS MULTIAGENTES (SMA)

La mayor parte de problemas que intenten solventarse con la tecnología de agentes podrían ser solventadas con un solo agente. Sin embargo, lo mismo que el concepto de programación descendente, modular y orientada a objetos intenta dividir los problemas en subproblemas cada vez más sencillos de resolver, en el ámbito de los agentes se huye de la idea del agente omnipotente. Es decir, lo mismo que en la sociedad humana podemos encontrar las actividades claramente divididas para conseguir un elevado grado de especialización y eficiencia, en el caso de los agentes se tiende a este planteamiento en el que múltiples agentes autónomos interactuan, se coordinan (cooperando, compitiendo o ambas cosas) y todo ello de una forma adaptativa. De esta forma se constituyen sociedades de agentes a las que se consideran Sistemas MultiAgentes (SMA).

Por tanto, aunque el uso de un solo agente aparezca como más simple en su implementación, ya que no es necesario prestar atención a las labores de coordinación, negociación, interacción, etc., parece más sensato abordar problemas de cierta escala, como es el caso de nuestra arquitectura de comunicaciones TAP, mediante un sistema multiagente, por, entre otros, los siguiente motivos:

- Aunque podemos tener un agente plenipotenciario, esto puede conducir a claros problemas de escalabilidad y pérdida de eficiencia, por lo que parece lógico dividir las labores y funciones que se desempeñarán entre varios agentes, consiguiendo así modularidad, flexibilidad, robustez, modificabilidad, escalabilidad y extensibilidad.
- El conocimiento concentrado en un solo agente puede conducir a una clara limitación de robustez del sistema, ya que si aparecen problemas en el agente que concentra todo el conocimiento, se propagan a todo el sistema. De este modo, es también intuitiva la ventaja de distribuir el conocimiento entre varios agentes que aporten un cierto grado de tolerancia a fallos.
- La distribución de la carga, de las tareas o de las funciones es otra de las ventajas que pueden obtenerse de los SMA. De este modo, los agentes de un SMA pueden diseñarse como componentes autónomos que actúan en paralelo para resolver problemas de ámbito general.

Como vemos, los agentes pueden ser entes autónomos o bien miembros de un SMA. Resulta relativamente sencillo extrapolar el concepto de agente hacia sistemas, poco o muy complejos, donde intervengan diversos agentes software coordinados entre sí y compartiendo informaciones que les sean de interés. De este modo podemos adquirir una idea clara e intuitiva de lo que sería un sistema o sociedad de agentes múltiples, distribuidos y cooperativos que interaccionan entre sí compartiendo información. Las ventajas aportadas por los SMA son muy variadas y algunas de las más interesantes descritas en la literatura han sido compiladas en [10] de donde destacamos y adaptamos a nuestras necesidades concretas las siguientes:

- Pueden usarse para enfrentarse a problemas excesivamente complejos para ser resueltos mediante agentes centralizados por la escasez de recursos o para lograr la optimización de éstos.
- Para ofrecer escalabilidad, ya que los agentes del SMA pueden evolucionar dinámicamente a medida que cambie o evolucione el propio sistema.
- Permiten o facilitan la interconexión con otros mecanismos clásicos, como son los sistemas expertos, los sistemas de ayuda a la toma de decisiones o, principalmente en nuestro caso, con pilas de protocolos de comunicaciones ya existentes.
- Para resolver problemas mediante soluciones provenientes de fuentes de información distribuidas.
- Finalmente, para conseguir las ventajas que todo sistema distribuido pueda aportar y, en suma, donde las características expertas del sistema puedan ser distribuidas en él mismo.

En cualquiera de los casos, los SMA ofrecen importantes ventajas sobre los sistemas de control centralizados, de la misma forma que la computación distribuida mejora las prestaciones de la centralizada.

Parece lógico que los agentes no actúan en solitario, sino que se localizan en entornos o plataformas con varios agentes. Cada uno de los agentes del entorno tendrá sus propios objetivos, tomará sus propias decisiones y podrá tener la capacidad de interactuar y comunicarse con el resto de agentes [42]. Los entornos

o plataformas que soportan varios agentes es lo que en la literatura se conoce con el nombre de SMA o agencias.

El OMG define un sistema multiagente como [32]: "*una plataforma que puede crear, interpretar, ejecutar, transferir y terminar agentes*". Como características principales de los SMA pueden destacarse las siguientes [15]:

- Cada agente individual del sistema tiene un conocimiento limitado del mismo, es decir, los agentes no disponen de información del sistema completo.
- No existe un control global de todo el sistema por lo que los datos quedan descentralizados.
- La computación es asíncrona en lo que respecta a la interacción entre los diversos elementos del SMA.
- Deben permitir la interoperación con otros sistemas existentes, aunque ésta es de las tareas más complejas en la actualidad.

Sin embargo, hay que destacar también que para poder soportar SMA es necesario disponer de un entorno adecuado para su desarrollo. Este tipo de entornos debe disponer al menos de las siguientes características:

- Contener agentes con las capacidades de autonomía, adaptabilidad, interacción y coordinación.
- Partir de un planteamiento abierto donde se huya de un diseño centralizado.
- Ofrecer una infraestructura donde se pueda especificar claramente la comunicación y los protocolos de interacción entre agentes y con el propio entorno.

Sin embargo [URL4], el diseño del escenario anterior suele plantear una serie de problemas como la elección del tipo de comunicación entre los agentes, del tipo de plataforma o del método de desarrollo, de los criterios para la seguridad del sistema, etc.

En todos estos casos podemos encontrar cómo las redes de comunicaciones son un claro ámbito de aplicación del paradigma de los agentes software distribuidos con sus múltiples variantes. Con la aplicación de las técnicas de agentes, las redes pueden ser más fácilmente administradas, sus recursos pueden ser optimizados ofreciendo QoS, los operadores de las redes pueden ver simplificado su trabajo, los usuarios finales pueden tener nuevos servicios y satisfacer sus necesidades de QoS y los proveedores de servicios pueden incrementar la productividad de sus redes. Éstas son algunas de las ventajas más generales, pero en el resto de este capítulo particularizamos en algunas de ellas.

6.4. COLABORACIÓN, COORDINACIÓN Y COMUNICACIÓN ENTRE AGENTES

Los problemas que intentan solventar los SMA se consiguen resolver en función del grado de cooperación o colaboración que los agentes individuales son capaces de compartir mediante una estrategia de interacción adecuada. Así, es muy importante poner un especial énfasis en el diseño de esta vital faceta de los agentes que tradicionalmente ha presentado tres estrategias de interacción básicas y que vamos a relatar resumidamente a continuación:

- Agentes cooperativos: Trabajan conjuntamente con la intención de resolver juntos un problema y pueden verse como un modelo adecuado para la gestión de redes. En general, esta estrategia es útil para el control de sistemas críticos donde debe conocerse el estado de equilibrio del sistema que, en nuestro caso particular, puede ser el control de errores de una red y sus congestiones. En conclusión, este tipo de agentes está pensado para sacrificar la utilidad individual a cambio del bien general de todo el sistema.
- Agentes auto-interesados son aquellos que intentan maximizar su propio beneficio sin preocuparse del interés general del sistema. Este planteamiento es útil en el caso de las redes cuando aparecen múltiples proveedores de red interesados en la bondad de sus propios servicios. Para nuestro ámbito de actuación puede tener interés aplicar este tipo de comportamiento en la negociación y mantenimiento de los parámetros de la QoS que los usuarios de la red pueden requerir. En general, este tipo de comportamientos son apropiados cuando los agentes van a competir en entornos abiertos, como pueden ser diferentes tecnologías de redes y necesidades de usuarios.

- Agentes hostiles: Este tipo de agentes no tiene una aplicación directa en el caso de los sistemas de telecomunicaciones y se basan en la idea de que tienen una utilidad que se ve incrementada con su propia ganancia y con la pérdida de otros agentes competidores.

Entre estas propuestas se han realizado múltiples investigaciones en el ámbito de la planificación distribuida, la planificación general, la asignación de recursos y el control de problemas mediante agentes cooperativos. Aunque el *Capítulo 7* profundiza en estos aspectos, queremos adelantar que algunas de las ventajas para la distribución son las siguientes:

- La reducción de los costes de comunicación asociados a los sistemas centralizados. Es importante destacar que, para que esta ventaja pueda ser conseguida, debe requerirse el mínimo consumo de ancho de banda para la comunicación inter-agentes.
- Incremento de la velocidad y optimización de cómputo a través de la paralelización.
- Incremento de la reactividad sin necesidad de consulta a entidades centralizadas.
- Robustez, disminuyendo las dependencias de un nodo concreto. En algunos casos esta robustez puede ser también conseguida mediante la replicación de algunos agentes, en lugar de la distribución.

Los agentes que forman un SMA deben colaborar entre ellos para desempeñar sus tareas respectivas. Esta colaboración suele realizarse mediante un lenguaje de comunicación comprensible por todos los agentes del sistema y también por otros programas si es necesario [URL4]. En contraste con un objeto convencional –que ejecuta métodos de otro objeto siempre que tenga permiso–, un agente puede rechazar una petición de otro agente, por lo que deben ser capaces de comunicarse entre sí, para decidir qué acción realizar o qué datos obtener [16]. El mayor inconveniente para conseguir la comunicación está en el hecho que existen múltiples lenguajes propios de cada fabricante, lo que no facilita la compatibilidad en sistemas heterogéneos.

A principios de los años 80, el DARPA estadounidense desarrolló el Knowledge Query Management Language (KQML) [URL5] como una norma para la comunicación entre agentes. Dicho lenguaje incluye múltiples primitivas –llamadas ejecutivas– que definen las operaciones que un agente realiza en su comunicación con cualquier otro. Además, un entorno KQML puede enriquecerse con agentes especiales –llamados facilitadores– que proveen funciones adicionales.

Algunos entornos –como el Java Agent Template (JATLite) de la Universidad de Stanford (EE.UU.) [URL6]– han desarrollado compatibilidad con KQML. Sin embargo, han aparecido varios dialectos KQML con una sintaxis similar, pero no completamente compatibles.

La Fundación para los Agentes Físicos Inteligentes (FIPA) [URL7] propuso como norma a principios de los años 90 el lenguaje Arcol (desarrollado por France Télécom [URL8]). Arcol conserva –al igual que KQML– una sintaxis similar al lenguaje LISP, pero incluye una semántica formal, lo que proporciona una base rigurosa para la interoperación entre agentes y evita la proliferación de dialectos [16]. Sin embargo, Arcol carece aún de algunos aspectos deseables en un lenguaje pensado para la comunicación de agentes, como permitir una mayor autonomía, una mayor heterogeneidad y el uso de dialectos abiertos.

Por otro lado, CORBA pretende definir una norma genérica para la comunicación y la interacción de sistemas multiagentes creados por distintos fabricantes.

Los agentes de un SMA deben coordinar sus actividades para alcanzar sus objetivos y ser capaces de negociar con otros agentes cuando aparecen problemas como la escasez de recursos. La coordinación es, por tanto, necesaria para determinar la estructura organizativa entre un grupo de agentes. Posiblemente, la técnica de coordinación más exitosa para la asignación de recursos y tareas entre una sociedad de agentes es Contract-Net Protocol (CNP) adoptado en las especificaciones de FIPA [URL4]. En CNP se aplica una estructura de mercado donde los agentes toman dos papeles principales, el de director y el de contratista. El principio base para la coordinación es que, cuando un agente no puede resolver la acción que tiene asignada usando su conocimiento local, lo que hará será descomponer el problema en subproblemas y buscará otros agentes con recursos suficientes que sean capaces de solventar los subproblemas obtenidos. La asignación de los subproblemas se solventa mediante un mecanismo de contratación por el cual el agente director crea un contrato que luego difunde (y éste puede ser un inconveniente en entornos de red) a otros agentes del sistema que pueden aceptarlo o no. CNP aporta las ventajas de permitir la asignación dinámica de tareas permitiendo la participación de otros agentes.

La comunicación entre los agentes y la forma de realizarla es otra de las características importantes de los SMA. Los agentes pueden pasar información a otros agentes para compartir entre sí los cambios producidos en el sistema o bien para informarse de los cambios previstos en el estado del sistema. Esto generalmente se realiza en los SMA a niveles elevados de abstracción, en lugar de usar los tradicionales métodos remotos de

paso de mensajes. Normalmente, la comunicación entre agentes implica el paso de información a nivel de conocimiento, y la comunicación puede realizarse a través de mecanismos dinámicos como es la memoria compartida, los IPC o a través de mecanismos estáticos como es el uso de ficheros con bloqueos. De este modo, la literatura presenta dos categorías de comunicación entre agentes; por un lado, mediante el uso de Agent Communication Languages (ACL) propietarios y por otro, mediante el uso de ACL normalizados.

La comunicación permite a los agentes de un SMA el intercambio de información con otros agentes para anunciar su situación actual, recursos disponibles, entorno local, etc. como base para cooperar con otros agentes. Mediante la comunicación de este tipo de informaciones los agentes pueden trabajar de una forma más flexible.

6.5. ARQUITECTURAS DE SMA

Tradicionalmente se ha presentado la arquitectura de agentes dividida en dos niveles generales. Por un lado, el primer nivel define la arquitectura interna del agente, es decir, las relaciones entre los componentes internos de cada agente. En segundo lugar lo que se define como la arquitectura del SMA que presenta las relaciones e interacciones entre cada uno de los agentes individualmente.

La referencia [5] presenta una clasificación de los agentes en función de la arquitectura interna del agente. Así, se pueden encontrar tres tipos de arquitecturas: reactiva, deliberativa e híbrida. Los autores de esta clásica referencia realizan una descripción detallada de estas tres arquitecturas así como de las investigaciones relacionadas con ellas. Por otro lado [9] analiza y discute los ámbitos en que se han aplicado estas arquitecturas internas, así como los campos en que podrán ser aplicadas en el futuro.

Existen otros planteamientos en lo referente al importante aspecto de las arquitecturas de agentes y el trabajo [13] presenta otras arquitecturas internas de agentes basándose en las técnicas de IA que pueden usarse como base de la tecnología de agentes. La *Figura 6.2* presenta una visión básica de agente donde pueden observarse las entradas, salidas y acciones realizadas por los agentes software para interaccionar con su entorno externo.

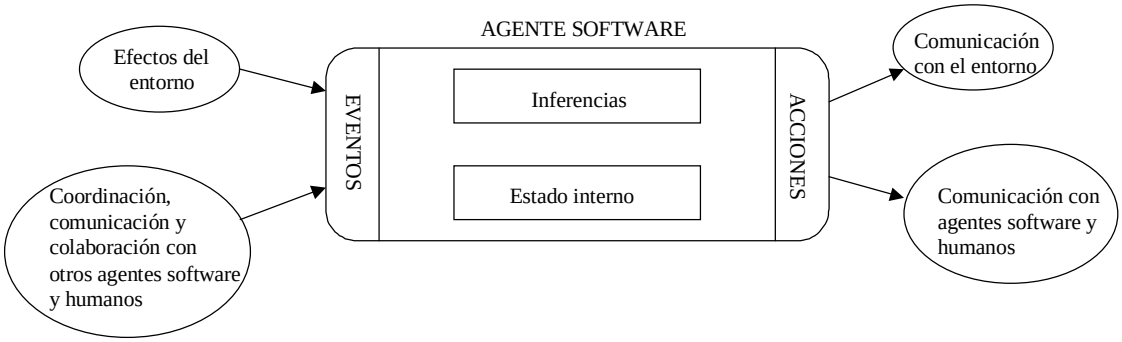

***Figura 6.2**. Interacción de agente software con su entorno*

En [10] se destaca que la adopción de una arquitectura interna de agente específica como oposición a otra arquitectura es inútil si la arquitectura multiagente no está bien definida. Así, el nivel superior de la arquitectura multiagente puede entenderse como un nivel que los agentes necesitan para poder coordinar sus mecanismos de colaboración en el SMA completo. Por tanto, en este nivel se pone especial atención en el grado de cooperación o colaboración que se facilita entre los agentes.

Existen en la literatura muy diversos modelos de SMA que acaban dando lugar a múltiples topologías que representan la situación de los agentes en la red, su interconexión y la forma en que resuelven sus tareas. En cualquiera de los casos suponemos el modelo de agente presentado en la *Figura 6.1*, de forma que estos agentes unidos en grupo constituirán el SMA, siguiendo dos modelos generales: la topología completamente mallada en que todos los agentes están comunicados entre sí, y la topología jerárquica en la que cada agente está solo conectado con su padre (predecesor) y su hijo (sucesor) en la jerarquía. La *Figura 6.3* muestra estas dos topologías en la que puede entenderse que cada agente únicamente se puede comunicar o interactuar con aquellos agentes con los que está conectado. Suponemos que cada uno de los nodos de la red soporta un agente software.

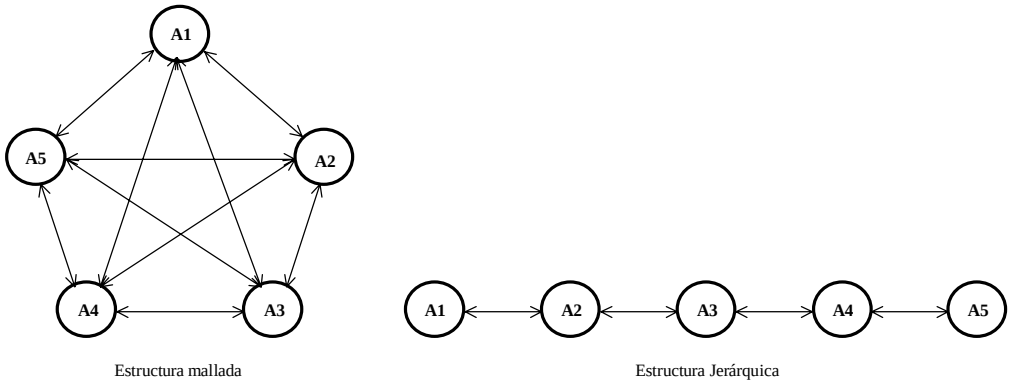

Figura 6.3. *Topologías generales de SMA*

La elección de un modelo mallado o jerárquico tiene importantes efectos sobre la escalabilidad, el rendimiento y la estabilidad del sistema. Estas tres propiedades no funcionales de los SMA son estudiadas en el *Capítulo 7*, sin embargo, podemos ver ya que es clara la implicación respecto a los aspectos de escalabilidad y conectividad entre los dos modelos que pueden observarse en la *Figura 6.3*. Si nos fijamos en el caso de una red completamente mallada en la que todos los agentes están conectados entre sí, puede calcularse el número de enlaces de una red de *N* nodos con la expresión

$$\frac{N(N-1)}{2}$$

Como en el caso de las redes de ordenadores, en el modelo mallado de agentes, el número de los nodos del sistema crece linealmente, sin embargo, el número de conexiones entre los nodos crece de forma cuadrática. Por ejemplo, si estamos ante un escenario que está formado por nodos con 6 agentes software cada uno, que deben estar todos completamente mallados, el número de agentes de la red viene dado por la expresión,

$$\frac{6(N(N-1))}{2} = 3N(N-1)$$

Así, la estructura jerárquica reduce la complejidad de computación limitando el número de sus agentes hijo y puede enfrentarse con el incremento de comunicación causado por la incertidumbre. La mayor ventaja es que el entorno puede ser escalable y flexible.

Es importante la consideración de los problemas de la escalabilidad en los sistemas multiagente, aunque éste es un aspecto poco estudiado y, menos aún en el caso de su aplicación a los sistemas de telecomunicaciones. Generalmente las investigaciones se han centrado más en las propiedades funcionales de los SMA donde se estudian las características de los agentes y la funcionalidades que son capaces de aportar. Las referencias [17,18] son dos interesantes estudios que se centran en la investigación de otros aspectos diferentes a los puramente funcionales, principalmente en las propiedades relacionadas con el rendimiento del SMA.

Podemos ver claramente cómo los modelos de SMA se ajustan perfectamente a su aplicación en los sistemas de telecomunicación, por lo que podemos pensar en extender estas dos topologías básicas a cualquiera de las topologías de redes que conocemos para dar lugar a nuevas estructuras de SMA. Por ejemplo, podemos pensar en estructuras de SMA en bus, en anillo, en estrella, en doble anillo y doble bus, en topologías arborescentes y en forma de grafos, en situaciones punto-punto, punto-multipunto y multipunto-multipunto donde cada nodo puede tener varios predecesores y múltiples sucesores, etc. Queremos destacar, por tanto, que podemos encontrarnos con múltiples variantes de los dos casos más comunes y que, sobre todo, hay que destacar la característica de posibilidad de comunicación en función de la conectividad que exista entre los agentes. Por extensión, por delegación o por cualquier otra vía, cualquier agente de un SMA podrá estar conectado con otro agente, lo mismo que cualquier nodo de una red puede comunicar con otro nodo de esa misma red aunque no exista una conexión directa entre ambos nodos. Como veremos, en nuestra propuesta de SMA-TAP estaremos ante una topología jerárquica arborescente para poder soportar las necesidades p-p, m-mp y mp-mp de las redes de tecnología ATM.

6.6. NORMALIZACIÓN DE LA TECNOLOGÍA DE AGENTES

En los últimos años, varios organismos (OMG, FIPA, CLIMATE, KQML, etc.) y proyectos de investigación auspiciados por los gobiernos (ESPRIT, DARPA, ACTS, CoABS) han intentado centrar los diversos aspectos de la interoperabilidad de los agentes, basándose en las siguientes características [19]:

- Debería ser posible acceder a un conjunto de servicios de infraestructuras compartidas, entrega de mensajes fiable, nombrado de agentes, interoperación estructural, etc.
- Debería ser posible compartir una ontología común, como teoría veraz y realista; y métodos de relación entre objetos y variables.
- Debería haber una convención en la sintaxis y semántica de un lenguaje de comunicación de agentes (ACL) común en el que sea posible expresar todo ello.

En cualquier caso, la mayor parte de las iniciativas de interoperabilidad coinciden en que estas tres normas parecen insuficientes, ya que existen múltiples plataformas de agentes que las cumplen pero que no son capaces de interoperar. Un primer problema aparece porque la interoperabilidad entre lenguajes no es suficientemente amplia. Específicamente, para el caso de ACLs como KQML [20] y el FIPA ACL [21], la interoperabilidad de los lenguajes requiere -más que simpleza- la convención de agentes en el formato y significado de las primitivas de los mensajes ACL. En cierto modo, se nota una carencia de especificaciones técnicas que faciliten la "conversación" entre diferentes técnicas de agentes.

En la última década se ha realizado una importante actividad investigadora en el área de la gestión de redes de telecomunicaciones. Muchas de estas investigaciones han culminado en TMN (Telecommunication Management Network) [22] con sistemas basados en los estándares implantados en la actualidad. Iniciativas más recientes como TINA (Telecommunicacions Information Networking Architecture) [23] han propuesto la gestión integrada con control de servicio para los teleservicios multimedia integrando y desarrollando los conceptos TMN y IN (Intelligent Network). Mientras TMN usa actualmente el modelo manager-agent y los protocolos OSI Systems Management para sus interfaces interoperables, TINA [24] aboga por el uso de tecnologías de procesamiento distribuido de propósito más general como es el OMB CORBA (Common Object Request Broker Architecture) [25].

En la actualidad, los entornos de agentes móviles están dirigidos por dos organismos preocupados por su estandarización. La FIPA (Foundation of Intelligent Physical Agents) [21] cuida de la interacción semántica de alto nivel entre los agentes software que desarrollan alguna forma de inteligencia y adaptabilidad, teniendo sus raíces en la IAD. OMG cuida más de la movilidad de acuerdo a un entorno interoperable estándar a través de su MASIF (Mobile Agent System Interoperability Facility) [26].

En los aspectos conceptuales y funcionales existe un elevado grado de coincidencia, sin embargo, en lo relativo a la normalización de plataformas de desarrollo, lenguajes de comunicación entre agentes, mecanismos de transporte de mensajes, ontologías, protocolos de integración, etc., nos encontramos ante una verdadera torre de Babel que intenta converger hacia una necesaria normalización de la tecnología de agentes. Los organismos más importantes en esta tarea normalizadora son los siguientes:

6.6.1. OMG

OMG (Object Management Group) [3] está constituido por varios Grupos de Trabajo que dirigen sus esfuerzos hacia la normalización de la tecnología de agentes. OMG publica periódicamente RFI (Request For Information) y RFP (Request For Proposal) solicitando opiniones y aportaciones en todo lo relativo a la tecnología de agentes.

6.6.2. FIPA

FIPA (Foundation for Intelligent Physical Agents) [URL7] está trabajando desde 1996 en el área de la estandarización de agentes y tiene múltiples trabajos y documentos desarrollados y en curso con muchas coincidencias con OMG. Sus mayores esfuerzos en la actualidad se centran en la creación de una arquitectura que pueda soportar múltiples mecanismos para la implementación de sistemas de agentes sobre sistemas de computación distribuida como son Java y CORBA.

6.6.3. CLIMATE

CLIMATE (Cluster for Intelligent Mobile Agentes for Telecommunication Environments) [URL9] representa un grupo de proyectos relacionados con la tecnología de agentes. CLIMATE se constituyó en

1998 en el ámbito de la Unión Europea y en el contexto del programa de investigación y desarrollo ACTS (Advanced Comunications Technologies and Services) y actualmente lleva a cabo un completo grupo de proyectos que investigan el uso de los agentes en varias áreas de aplicación como es el control de servicio en redes móviles y fijas, la gestión de telecomunicaciones, el comercio electrónico, las aplicaciones multimedia, etc. En este caso, se incorporan las especificaciones estándares de FIPA y OMG.

6.7. VENTAJAS DEL USO DE AGENTES EN COMUNICACIONES

A pesar de la gran actividad desarrollada en este campo en los dos o tres últimos años, la tecnología de agentes está aun hoy en un estado casi embrionario. Su ámbito de aplicación es tan amplio como el de la programación en general. Sin embargo, hemos de destacar que la aplicación de los agentes al campo de las comunicaciones ha sido y es de los más activos. De este modo se han propuesto agentes para asistir o ayudar en labores complejas de administración de redes[7], para el balanceo de la carga, para anticiparse a los fallos de los sistemas de telecomunicación, para analizar problemas o para sintetizar todo tipo de informaciones relativas a diversos aspectos de las redes. Destacable es el hecho de la rápida evolución de las comunicaciones y computación móvil, donde los agentes -especialmente en su característica de movilidad- aportan importantes propiedades.

Los sistemas multiagentes suelen ser protocolos del nivel de aplicación situados sobre una capa de distribución (ej. CORBA) que ofrecen varias características destacables como son:

- Un nivel de comunicación en la capa de aplicación que es común a todos los agentes.
- Protocolos de interacción que describen cómo los agentes pueden coordinar sus actividades separadas. Los agentes pueden “negociar” para decidir cómo deben actuar.
- Escalabilidad: permitiendo la expansión de una red añadiendo más autoridades en lugar de incrementar la capacidad de los existentes.
- Efectividad: distribuyendo y reduciendo la gestión del número de saltos desde los extremos hasta los nodos activos (que soportan agentes software), lo que minimiza las latencias en la comunicación (QoS ATM).

La coordinación entre nodos activos de la arquitectura propuesta aporta beneficios específicos al control de tráfico en las redes ATM y, entre ellos, podemos destacar:

- El uso más eficiente de recursos (los agentes los optimizan).
- La naturaleza distribuida y adaptativa de los agentes aporta redes más flexibles y escalables.
- Alivian el trabajo de los operadores humanos.
- Los agentes toman decisiones de forma estándar, lo que reduce las dependencias de la integración de fabricantes.
- Pueden tomarse decisiones de conexión en QoS y en otros factores como la caracterización del tráfico en coste.
- Redes más fiables por el control distribuido aportado por los agentes (la arquitectura TAP que proponemos aprovecha especialmente esta característica).

En nuestro caso, con la arquitectura TAP proponemos un nuevo paradigma para la inclusión de agentes software programables en los conmutadores activos ATM. Los agentes que proponemos se ejecutan autónomamente y cooperan coordinadamente todos entre sí para obtener un objetivo general que es la consecución de una red ATM activa programable. Concretamente, nos centramos en un problema puntual donde los agentes se encargan de solventar una situación que las propuestas estándares no resuelven como son las pérdidas de células en situaciones de conmutadores congestionados.

[7] En el ámbito de las redes TCP/IP el concepto de agente SNMP se ha usado durante años para la realización de labores de gestión de red, pero este tipo de agente sólo coincide en el nombre con los agentes que estamos tratando.

6.8. GESTIÓN CENTRALIZADA Y DISTRIBUIDA DE REDES MEDIANTE SMA

La IAD y la aplicación de conceptos y tecnologías de agentes inteligentes puede ser también usada para unificar las ventajas de la gestión de redes centralizada y distribuida. La IAD promueve el uso de comportamientos más que el control funcional. De este modo, los agentes pueden ser usados para la construcción, mantenimiento y comprensión de sistemas grandes y complejos. En esta línea el trabajo [27] propone un sistema jerárquico de controladores independientes (agentes) con solución de problemas locales y características de construcción de decisiones. Cada agente actúa como sistema de gestión centralizado, pero colectivamente se comunican entre sí para mantener objetivos generales del sistema y para resolver conflictos de interés entre todos ellos. Así pueden obtenerse arquitecturas híbridas aplicables a la gestión distribuida y al control de tareas como la gestión de redes heterogéneas, la gestión de QoS y la administración de administradores de redes.

La asignación de recursos es un problema difícil de resolver en la gestión de redes, principalmente porque el estado operacional de éstas es demasiado dinámico, sobre todo en las redes de banda ancha como ATM. Muchas investigaciones se han dedicado a las estrategias óptimas para el control de admisión de conexión, rutas topológicas, configuraciones de capacidad óptima. Sin embargo, existen muy pocas investigaciones relacionadas con el problema de la reconfiguración topológica que requiera la coordinación de diversos aspectos de la red. Una de las investigaciones en el ámbito de ATM es [28], que usa un agente *peer-to-peer*. Contrastando con lo anterior [29] y [30] proponen una descomposición jerárquica del sistema multiagente y [6] propone un SMA estratificado en múltiples capas.

Otro interesante trabajo es [31], que presenta un agente cooperativo para enfrentarse al problema del control de congestión en las redes ATM. Usa agentes localizados en los buffers de los conmutadores para ofrecer información de realimentación al agente del nodo fuente para que las conexiones que sean añadidas después a la red no se incorporen a la zona congestionada.

Por otro lado, FIPA [URL7] ha definido los estándares para la aplicación de agentes en la gestión de redes. El escenario particular de la especificación usa agentes cooperativos especializados en la negociación para generar una red privada virtual (VPN). Este escenario presenta tres niveles: un agente para la representación de cada usuario humano, conocido como Personal Communications Agent (PCA). El segundo tipo de agente es el Service Provider Agent (SPA) que representa el interés y objetivos del proveedor de servicios y, por último, el Network Provider Agent (NPA). Los PCAs negocian con los SPAs existentes para determinar el mayor nivel disponible en materia de QoS y coste.

Aunque la asignación de recursos es uno de los factores más importantes en la aplicación de la tecnología de agentes en las redes de comunicaciones existen otras propuestas como las siguientes:

- Comunicaciones de un agente con el resto de agentes, con la intención de minimizar esa intercomunicación y hacerlo lo antes posible.
- Estudio de la situación donde deben situarse los agentes, por ejemplo, un agente por nodo, un agente por enlace físico, uno por cada conexión, etc.
- Análisis sobre si los agentes deben ser estáticos o móviles.
- Visión que los agentes pueden tener del resto de la red. Es decir, determinar si los agentes pueden tener o no conocimiento de los eventos locales a su entorno.
- Estudio del grado de dependencia que debe existir entre los agentes de un SMA. Esto permite determinar la robustez del sistema y saber cómo comportarse en el caso de que un agente concreto deje de actuar.

6.9. REVISIÓN DE AGENTES SOFTWARE APLICADOS A LOS SISTEMAS DE TELECOMUNICACIÓN

El trabajo [10] presenta resumidamente los proyectos más interesantes en materia de agentes software en el ámbito de los sistemas de comunicaciones. Cabe destacar que tradicionalmente se han clasificado también los agentes en cuatro categorías principales que son: los basados en SMA; los que emplean la tecnología de agentes móviles; los entornos basados en hormigas y, por último, las propuestas basadas en técnicas económicas.

Se puede decir que las iniciativas en la aplicación de agentes a los diferentes aspectos de las telecomunicaciones, aunque no abundantes, si empiezan a ser significativas. En este apartado comentamos algunas de estas propuestas sin ánimo de ser extensivos ni tampoco profundos en la descripción.

6.9.1. GESTIÓN DE SISTEMAS DE COMUNICACIÓN MEDIANTE AGENTES MÓVILES

El OMG realiza en MASIF [URL7] las siguientes definiciones:

- Agente estático: aquél que sólo puede ejecutarse en la máquina donde fue iniciado. Si éste necesita interactuar con otros agentes o programas o requiere cierta información que no se encuentra en el sistema, la comunicación puede llevarse a cabo mediante cualquier método de interacción para objetos distribuidos, como CORBA o RMI de Java.
- Agente móvil: aquél que no está limitado al sistema donde se inició su ejecución, siendo capaz de transportarse de una máquina a otra a través de la red. Esta posibilidad le permite interactuar con el objeto deseado de forma directa sobre el sistema de agentes donde se encuentra dicho objeto. También puede utilizar los servicios ofrecidos por el sistema multiagente destinatario. Ningún sistema de objetos distribuido existente en la actualidad necesita utilizar agentes móviles para comunicarse.

Los agentes móviles suelen programarse normalmente en lenguajes interpretados o generadores de código intermedio –Telescript [URL10], Java [URL11], Tcl [URL12]–, ya que éstos dan un mejor soporte a entornos heterogéneos, permitiendo que los programas y sus datos sean independientes de la plataforma utilizada.

El proceso de transferencia de un agente de un sistema a otro suele realizarse en tres etapas [32]:

1. Iniciación de la transferencia.
   - El agente identifica el destino deseado, realiza una petición de viaje al sistema y –si es aceptada– recibe el permiso para ejecutar la transferencia.
   - El sistema suspende la ejecución del agente e identifica el estado y las partes del agente que serán enviadas.
   - Se realiza la conversión en serie del código y del estado del agente (serialización) y se codifica según el protocolo establecido.
   - El sistema hace la autentificación del agente.
   - Se realiza la transferencia.
2. Recepción del agente.
   - El sistema destinatario acredita al cliente.
   - Se realiza la decodificación del agente y la conversión en serie del código y estado del agente (deserialización).
   - El sistema crea la instancia del agente, restaura su estado y continúa la ejecución.
3. Transferencia de otras clases (sólo en sistemas orientados a objetos).
   - Este proceso es necesario cuando el agente se mueve de un sistema a otro, cuando el agente se crea remotamente, o cuando necesita otros objetos. La transferencia de las clases puede realizarse completamente junto con el viaje del agente o hacer peticiones de carga cuando sea preciso.

En cuanto a la interoperación entre sistemas multiagentes cabe destacar que la normalización en el proceso de interconexión de agentes móviles suele aplicarse a dos niveles:

- Interoperación entre diferentes lenguajes de programación.
- Interoperación entre sistemas escritos en el mismo lenguaje.

La primera de ellas resulta muy compleja de alcanzar y continúa aún en proceso de estudio, mientras que el segundo caso está siendo normalizado por CORBA. No obstante, para alcanzar el grado de interconexión deseado, las MASIF de CORBA definen los siguientes conceptos destacables:

- Lugar: contexto dentro de un sistema donde puede ejecutarse un agente; por lo tanto, un agente viaja de un lugar a otro, ya sea en el mismo sistema o en otro distinto.
- Localización: va asociada con un lugar, indicando el nombre y la dirección que ocupa en el SMA.

- Localidad: propiedad de cercanía al destino, ya sea en el mismo ordenador o en la misma red.
- Infraestructura de comunicación: provee servicios de transporte al sistema.
- Autoridad: persona o entidad en nombre de la cual actúa un agente o un sistema de agentes.
- Región o dominio: conjunto de sistemas de agentes que tienen la misma autoridad y que no tienen que ser del mismo tipo.

Los agentes software móviles ofrecen una forma radicalmente diferente de controlar sistemas distribuidos con algunas ventajas clave como la simplicidad, robustez o el tamaño del código resultante. Los agentes móviles son un tipo de aplicaciones beneficiosas donde la robustez y la habilidad para la autoregulación son más importantes que el tiempo de respuesta. Pueden ser muy apropiados para llevar a cabo tareas de ajuste de parámetros en la gestión de sistemas. Este atributo permite a los agentes ejecutar tareas mundanas que sólo pueden ser realizadas por supervisores humanos.

Este tipo de agentes ha recibido una especial atención en los últimos años debido principalmente [10] a la proliferación de tecnologías para la integración de plataformas como Java RMI, CORBA [33], *middleware* y a la relativa disponibilidad de hardware potente. De esta forma se desea aportar nuevos mecanismos que permitan actualizar dinámicamente los procesos de control en redes de comunicaciones y en esta línea se han realizado dos importantes propuestas como es, por un lado, la tecnología de Redes Activas tal como presentaremos en el siguiente apartado, y, por otro, el uso de tecnología de agentes móviles que aporta la movilidad del código dentro de las redes.

Los agentes móviles se plantearon desde su inicio para aportar la movilidad del código ejecutable entre ordenadores. La idea básica es aprovechar los recursos de un ordenador cuando los de otro son escasos. Aunque intuitivamente la idea de la movilidad de los agentes parece bastante sencilla, la verdadera labor de investigación está en determinar la forma en que podemos mover la llamada inteligencia de los agentes de unos nodos de la red a otros.

En muchos casos quizás sea más eficiente el paso de mensajes tradicional que la utilización de código móvil de unos nodos a otros, sin embargo, son muy atractivas las posibilidades ofrecidas por las agentes móviles para aportar mecanismos útiles para situar dinámicamente nuevos controles software en entornos de redes. Los agentes móviles aportan la característica de no necesitar una conexión continua que conlleve el consumo de recursos de red. Además, este tipo de agentes pueden ser creados fuera de la red y cargados en la misma para realizar las tareas que tengan encomendadas por el usuario que puede invocar al agente cuando lo desee. Otra ventaja de los agentes móviles respecto al paso de mensajes se da cuando los ordenadores a comunicar están muy separados geográficamente que suele ser lo habitual en los entornos de comunicaciones. Además, en la mayor parte de situaciones, el problema no puede solucionarse únicamente con un paso de mensajes, sino que se requiere el envío de piezas de software, que mediante el uso de agentes es más ágil que a través del paso de mensajes. La literatura describe todas las ventajas aportadas por el uso de este tipo de agentes y la referencia [6] presenta una visión diferenciada entre agentes software y agentes móviles.

El uso de agentes móviles combina algunos de los beneficios del control centralizado y distribuido. En cada etapa de los algoritmos los agentes pueden aprender [34] sobre el crecimiento de la red interrogando a un nodo para descubrir su estado y el estado de los enlaces con sus vecinos. No es necesario almacenamiento de información centralizada porque no se mantiene información local en los agentes. Pueden fallar agentes individuales, mientras el sistema general continúa funcionando. Hay algoritmos que son muy complejos de implementar en una arquitectura completamente distribuida, sin embargo, son más simples si se aprovechan algunas de las ventajas del control distribuido.

Los agentes móviles aparentan ser distribuidos en el sentido que no hay controlador central. Sin embargo, los agentes tienen una visión del sistema distribuido que no es local y, desgraciadamente, puede ser global porque, en principio, un agente móvil podría visitar cada parte del sistema distribuido para capturar datos. Igual que los agentes tienen las ventajas de los sistemas centralizados, inherentemente padecen también los inconvenientes de estos sistemas. Las decisiones en el control centralizado pueden estar basados en datos anticuados, porque lleva cierto tiempo mantener todo el estado del sistema. Como los agentes deben obtener la información de las diferentes partes del sistema, puede ser que lo hagan de aquellas que aún no tienen los datos actualizados. El envejecimiento de los datos que usa un agente puede ser controlado restringiendo el número de nodos que un agente puede visitar para captar datos. Si el número de nodos visitados por un agente es pequeño, puede que sufra por un excesivo control general del sistema. Sin embargo, esta forma de actuación permite balancear las características entre control distribuido y centralizado.

Los agentes presentan comportamientos algorítmico y heurístico [35]. Tienen comportamiento algorítmico cuando calculan unas tablas de routing, y comportamiento heurístico cuando gestionan su carga de trabajo eligiendo qué agentes lanzar, terminar por si mismos o moverse a otro nodo. Éste es un poderoso argumento para introducir un cierto grado de inteligencia en un sistema puramente algorítmico.

Los agentes software móviles son procesos computacionales con la capacidad del movimiento y basados en principios organizativos observados en insectos sociales como las hormigas. La aspiración es conseguir la emulación de esos principios para lograr trasladar a los sistemas de comunicaciones características de los sistemas naturales como la simplicidad, la adaptabilidad y la robustez.

A medida que crece la complejidad de los sistemas de telecomunicaciones, crecen también las debilidades del control completamente distribuido o centralizado. Los sistemas con control centralizado tienen problemas de escalabilidad con el tamaño del sistema, debido al incremento de comunicaciones y procesamiento, y son potencialmente vulnerables a fallos de control. En control distribuido, los procesos se ejecutan en procesadores diferentes. Los procesos distribuidos necesitan coordinarse entre sí para identificar que se ha producido un evento que requiere control. La decisión de la acción de control que hay que tomar es dependiente de las habilidades de los procesadores para cooperar.

Son necesarias nuevas formas para organizar los sistemas de control para aliviar su complejidad. Parece deseable que pudiesen unirse características de los sistemas centralizados y distribuidos en lugar de tener que decidirse por uno concreto. De este modo, necesitamos [35] control software que sea:

- Intrínsecamente robusto a los fallos del sistema y de programas.
- Intuitivo para ser escrito de una forma bien estructurada.
- Autoregulador.
- Requiera la mínima cooperación directa entre procesos.

Las características anteriores, requeridas para los sistemas de control, se dan desde hace años en las hormigas. Una colonia de hormigas es claramente un sistema distribuido. Sin embargo, las hormigas son móviles, todo lo contrario a los agentes de un sistema distribuido convencional. Observando la forma de localización de comida en las colonias de hormigas se observó:

- Las hormigas son móviles.
- Los mensajes que intercambian entre sí son estacionarios.
- Los mensajes no son dirigidos a una hormiga específica.
- Los mensajes se atenúan con el paso del tiempo.
- Algunos de sus comportamientos son aleatorios.
- Existen un gran número de sencillas hormigas en lugar de unas cuantas complejas.

Precisamente las anteriores características forman la base de la mayoría de los agentes móviles. La movilidad es conseguida permitiendo a los agentes transmitirse para continuar trabajando en otros procesadores del sistema donde se desarrollan. Cada vez que un agente se mueve a otro procesador diferente deja un mensaje estático que puede ser leído por otro agente que venga detrás de él. Esto permite que otros agentes puedan trabajar adecuadamente. Éste es un procedimiento muy adecuado para sustituir agentes erróneos por nuevos agentes para compartir una tarea cuando los recursos son insuficientes. La atenuación de los mensajes se consigue con un *time-stamp* que va haciendo que los mensajes sean poco a poco menos relevantes. De este modo, los mensajes erróneos, o que han perdido importancia, pueden ser sustituidos del sistema. El comportamiento aleatorio también puede ser introducido en los agentes. En un sistema realmente complejo el número de excepciones que necesita un agente para cubrir cada situación puede ser inabordable, por lo que puede ser de interés permitir a los agentes solventar situaciones mediante elecciones aleatorias que no tienen por qué ser previamente preprogramadas en los roles del agente. Por último, si podemos diseñar un sistema de control de agentes sencillos, éste será más pequeño, más fiable y fácil de programar.

El trabajo [35] presenta un sistema de control basado en un agente móvil, que cumple con una disciplina de programación que está bien estructurada y es robusta ante el fallo de un agente o de un componente del sistema distribuido. Para ello se proponen las tres siguientes características en los agentes para que sean intrínsecamente robustos:

- No debería haber intercomunicación directa entre agentes. Si no existe esa intercomunicación, un agente no podría enviar un mensaje cuyo destino sea una instancia específica de otro agente. Por tanto,

un agente no debería enviar ni recibir un mensaje desde una instancia específica de otro agente. Esto aporta a los agentes móviles autonomía e independencia, que es una importante característica para la robustez. Sin embargo, los agentes deben disponer de algún mecanismos para que puedan pasarse los resultados de sus actividades entre sí. La forma de hacer esto es a través de la modificación del estado del sistema que controlan los agentes. Este tipo de comunicación sólo es posible si los agentes son móviles.

- Los agentes deberían ser un número razonablemente alto. De este modo, a lo que se aspira es a que entre pocos agentes soporten una importante proporción de la responsabilidad de control del sistema. La reunión de agentes será intrínsecamente robusta al fallo de un agente individual. Por supuesto, el fallo de un agente causará una cierta degradación del sistema, pero será gradual en lugar de catastrófica. La gradual, en lugar de repentina, degradación del rendimiento es una de las ventajas más significativas del uso de agentes móviles.
- Los agentes deberían ser capaces de alterar dinámicamente sus tareas de asignación y su número. Esta tercera propiedad implica que el número de agentes debería ser proporcional a la carga de trabajo y que los agentes que fallen podrán ser detectados y reemplazados. Los agentes fallidos deben ser detectados por el conjunto de agentes que deberán compensar los errores incrementando el número de agentes al nivel previo al fallo.

El mecanismo propuesto para lograr lo anterior es el uso de *time-stamp* en los elementos de datos que son modificados por los agentes como parte de su función normal cuando visitan un nodo. Éste es un dispositivo muy sencillo pero con un potencial importante para permitir la unión de los agentes para su auto-organización. Los agentes cuando se mueven por la red pueden juzgar si una tarea tiene o no recursos mediante la edad de los mensajes encontrados. Es decir, si un agente encuentra mensajes recientes relativos a su tarea actual, podrá asumir que la tarea estará siendo realizada por otro agente.

El flujo de tráfico en la red es simulado por un módulo generador de tráfico. Un perfil de tráfico determina los nodos fuente y destino, tiempo de inicio, duración y ancho de banda de las conexiones que van a añadirse a la red. Este control de sistemas de agentes móviles puede ser entendido como un ecosistema de entidades de control. Para añadir nuevas funciones de control, puede crearse un nuevo tipo de agente, dando mecanismos para interactuar y gestionar su población y después crear el ecosistema. Esto hace el sistema altamente extensible.

El artículo [36] presenta el uso de agentes móviles para distribuir y automatizar las funciones de gestión de red en ATM. La gestión a través de agentes móviles ofrece un paradigma para la gestión de red distribuida, flexible, escalable y robusta que mejora las limitaciones de los actuales esquemas de gestión centralizada. Los sistemas de gestión de red se basan en un paradigma trabajo-intensivo con plataforma de inteligencia centralizada y aplicada a datos distribuidos. Los datos operativos son capturados por agentes empotrados en dispositivos de red y recogidos por una plataforma centralizada usando un protocolo de gestión como SNMP o CMIP. La centralización presenta conocidas limitaciones como la escalabilidad y la adaptabilidad de la gestión. Es decir, cuando el tamaño (número de dispositivos), complejidad (número de variables gestionadas), o la velocidad de la red crece, o si las velocidades de gestión de la red son limitadas, el sistema de gestión centralizada se convierte rápidamente en inmanejable. Para solventar estos y otros problemas de la gestión centralizada se han propuesto múltiples soluciones descentralizadas [37-40]. Es importante comentar también que la gestión descentralizada ha sido seguida de lo que se ha denominado gestión por delegación [41], que aprovecha la movilidad del código.

La tecnología de código móvil suele dividirse en código-bajo-demanda, evaluación remota y agentes móviles [42]. Mientras en código-bajo-demanda y evaluación remota la atención se fija en la transferencia de código entre componentes, en el paradigma agente móvil se mueve a un lugar remoto un componente computacional completo junto con su estado (el código que necesita) y algunos de los recursos necesarios para realizar las tareas a desempeñar.

El artículo [36] presenta una propuesta de sistema de gestión para conmutadores ATM con un paradigma basado en agentes móviles que implementan las funciones de gestión de fallos y rendimiento. Como la gestión de fallos puede ocurrir en los periodos duros de congestión, la función de gestión es bastante importante, aunque la función de gestión de rendimiento pueda sobrecargar el tráfico de la red de forma importante con respecto a otras funciones. Por esta razón estas labores se realizan esporádicamente para no sobrecargar innecesariamente la red.

6.9.2. GESTIÓN DISTRIBUIDA DE FALLOS EN REDES ATM MEDIANTE AGENTES

En la referencia [43] se presentan varios problemas que son solventados mediante agentes. Este trabajo demuestra que la comunicación entre los agentes es, o puede ser, el verdadero cuello de botella de esta propuesta que obtiene muy buenos resultados en aplicaciones sobre la gestión de fallos sobre redes locales ATM. Este trabajo propone el uso de agentes mediante un sistema distribuido dividido en tres grandes módulos: las fuentes de conocimiento, el módulo caja-negra y el de mecanismos de control. Es destacable que esta propuesta propone una arquitectura con un paradigma distribuido pero situado en capas superiores de la tecnología ATM. En realidad, la propuesta está situada por encima de la capa AAL-5, lo que acaba redundando en limitaciones en su rendimiento.

6.9.3. GESTIÓN DE REDES ATM MEDIANTE SMA

La referencia [6] describe la metodología software adoptada para la construcción de un sistema de gestión de red basado en un sistema multiagente. El proyecto Tele-MACS [URL13] es una aproximación al problema de la configuración (asignación de rutas y de ancho de banda) lógica de recursos de red dinámicamente adaptada a la utilización de los usuarios de la misma. Esto se consigue con técnicas de control basadas en el uso de un SMA distribuido. Se presenta la arquitectura de control multiagente distribuido. Se acoplan las características de planificación de los tipos de agentes con las actividades de agentes reactivos, para que el sistema de gestión de red pueda adaptar la configuración de recursos para cumplir con los cambios en la solicitud de conexión de los usuarios.

En las redes ATM la importancia del control de red fue situada inicialmente en el desarrollo de algoritmos que pudiesen estimar la utilización del ancho de banda para ciertos tipos de conexión a elevados grados de probabilidad. Una razón para esto fue que, en sistemas de comunicación de alta velocidad, las estrategias de control trabajasen en escalas de tiempo más pequeñas que las del plano de gestión. Sin embargo, en redes reales, estas funciones de control no actúan independientemente, pero son parte de una estrategia general. De este modo, la aspiración de la gestión de recursos de red es ofrecer al cliente la QoS requerida, a la vez que se permite al operador de la red usar ésta de una forma económica y eficiente. Aunque ATM fue concebida como una tecnología para satisfacer la reconfiguración dinámica de recursos de red en tiempo de ejecución, uno de los mayores problemas es la complejidad resultante. Una de las formas de solucionar este problema es distribuir el control y permitirle a agentes especializados la gestión de tareas concretas. Otra forma de reducir la complejidad resultante ha sido el control jerárquico y la planificación de comportamiento pro-activo. Tele-MACS explota la potencia de ambos métodos.

Considerando la unión entre planificación y reacción en los sistemas de control, se ha desarrollado un entorno de agentes multi-nivel que contempla el aumento de características de agentes de planificación y reactivos, cooperando en un mundo común. La idea está basada en la construcción de un sistema de gestión inteligente desde múltiples capas. Tele-MACS se basa en capas porque permite aislar o encapsular los agentes interactuando con un cierto entorno que aporta extensibilidad (creando interfaces bien definidas entre las capas) y que ofrece robustez a los fallos de software y robustez contra la imposibilidad de realizar una secuencia de acciones en el intervalo de tiempo del ciclo de control (las capas permiten operar en diferentes escalas de tiempo). Primero se construye el sistema coordinado completo que consigue el propósito específico a un cierto grado de competencias. Después se testea el sistema y se realizan ajustes hasta que se considera que el sistema satisface los objetivos previstos. Finalmente se construye otra capa completa de control a alto nivel de competencia y se ajusta a un mecanismo de supresión.

Algunas de las ventajas del uso de procedimientos tradicionales de control de admisión de conexión (CAC) son:

- En general, el algoritmo de CAC debe ser ejecutado en cada conmutador de una ruta candidata. Esto es, cada conmutador individual determina si puede soportar o no la llamada en un modo local. Esto implica un cierto *overhead*, especialmente en casos en que la función de conexión sea rechazada en alguna ocasión a lo largo de toda la ruta.
- La mayor parte de los algoritmos CAC incorporan simplificaciones, y no se pueden usar para el rango completo de servicios que pueden ser soportados por las redes ATM, ya sea en la actualidad o en el futuro.
- Varios algoritmos CAC eficientes, que intentan explotar al máximo las ventajas de la multiplexación, no pueden ser implementadas en tiempo real.

Son conocidos los variados problemas que se encuentran en la construcción de sistemas software para la resolución de problemas grandes y complejos. El proceso completo de diseño, modelado e implementación

requiere un importante esfuerzo, tiempo y dinero. Aparecen así las ventajas conceptuales ofrecidas por la programación orientada a objetos (POO) que son bien conocidas; sin embargo, el diseño de SMA puede ser visto como una extensión del diseño OO, especialmente cuando muchos investigadores consideran el diseño de SMA como una metáfora más abstracta.

En Tele-MACS primero se describen las bases de cada agente y algunas de las funciones que realizan. Después se ofrecen detalles de sus interacciones en el contexto de control de comunicaciones.

Se describe una demostración intuitiva de cómo se ha usado la idea de crear un sistema de gestión multiagente que pueda controlar selección de rutas de ancho de banda dinámico en redes ATM para evitar las congestiones. Un agente de CAC (CAC Agent) decide sobre las siguientes opciones:

- Hay suficiente ancho de banda disponible y la conexión puede ser aceptada.
- El ancho de banda es insuficiente, pero la conexión puede ser aceptada si se modifican los VPs usados.
- Hay insuficiente ancho de banda, pero puede negociarse con otro proveedor de servicio para obtener más.
- No se podrá obtener el ancho de banda y la solicitud de conexión es rechazada.

Hay dos tipos de agentes de usuario: el PUA (Proxy User Agent) y los PCA (Proxy Connection Agents) que tienen una funcionalidad similar. La diferencia es que el PUA de flujo ascendente conecta a una aplicación llamante y el PUA de flujo descendente a una aplicación llamada, mientras el PCA es la interfaz a solicitud de conexiones ascendente o descendente desde otra red. El PUA colecciona los parámetros de los mensajes de señalización y después comunica con el PUA destino para investigar cuándo el sistema final destino acepta la solicitud de conexión. El PUA envía después un mensaje al CACA del nodo fuente negociando los parámetros de la solicitud de conexión. El Agente CAC interroga a los posibles RA (Resource Agents) que sean capaces de ofrecer los requerimientos de recursos específicos. El Agente CAC toma las respuestas y decide el RA preferido para establecer la conexión. El RA elegido interactua con los agentes Switch Wrapper (SwWA) para hacer las conexiones cruzadas en cada conmutador a través del *path* seleccionado. Los SwWA implementan una API que mapea los mensajes de los agentes a los protocolo de señalización ATM. El agente SwWA constituye una abstracción software virtual de un conmutador ATM y sus recursos, y ofrece una interfaz genérica independiente del vendedor para control de red y aplicaciones de administración.

6.10. REDES ACTIVAS

Desde sus inicios, las redes han dispuesto de una gran variedad de elementos hardware (*switches*, *routers*, *bridges*, *brouters*, *hubs*, sistemas finales, etc.) con funciones bien conocidas (conmutación, enrutamiento, puenteo, control de congestión y de flujo, garantía de QoS, ejecución de aplicaciones, etc.). Las redes actuales son canales de comunicación que transfieren paquetes entre sistemas finales usando el hardware citado anteriormente. Pero también existen nuevas investigaciones para equipar los elementos hardware con elevadas prestaciones mediante técnicas software. Esto permite equipar la red con características activas (*activenets*[8]) en las que los elementos hardware computan, cambian y operan los paquetes y también transfieren y propagan código. Así, una red activa es una red programable que permite que el código sea cargado dinámicamente en los nodos de la red en tiempo de ejecución. La literatura sobre redes activas [28,44-57] estudia varios mecanismos para aprovechar los nodos activos. Sin embargo, las propuestas son insuficientes para las redes de tecnología ATM y la referencia [28] es un ejemplo de estas investigaciones para ATM.

Las redes IP basadas en protocolos estandarizados *de facto* se encuentran con la situación de tener que operar sobre muy diversas redes físicas actuando a modo de capa de red virtual encima de estas redes. Ante esto se realizan propuestas para conseguir redes más adaptables, tanto a la red física, como a las aplicaciones que se estén usando sobre ellas. Como IP no dispone de la habilidad de adaptarse a las necesidades de cambios, en las aplicaciones se propuso [45] un nuevo tipo de arquitectura de red llamada red activa. Esta arquitectura permite a las aplicaciones extender sus funcionalidades de una forma dinámica inyectando protocolos personalizados, también conocidos como *protocolos específicos de las aplicaciones*. En este tipo de redes los paquete eligen el protocolo por el que desean ser procesados.

[8] La noción Active Network representa la idea que los nodos de las redes de comunicación pasan a ser activos por tomar parte en el procesamiento de las aplicaciones y en el soporte de servicios personalizados.

No hay consenso para decidir cuándo una red es activa, pero existen dos grandes tendencias:"*...una red es activa si incorpora nodos activos con la capacidad de ejecutar programas de usuario.*" , o bien si ésta "*...implementa mecanismos de propagación de código entre los dispositivos de interconexión* (conmutadores ATM en nuestro caso)". La arquitectura TAP es activa en ambos sentidos porque sitúa nodos activos en puntos estratégicos que implementan un protocolo activo que permite que el código de usuario sea cargado dinámicamente en los nodos de la red en tiempo de ejecución. También se ofrece el soporte de propagación de código en la red gracias a las células RM. TAP es también una arquitectura distribuida en el sentido que el protocolo usa varios agentes coordinados y auto-colaborativos entre conmutadores activos.

Las redes activas, abiertas y programables son por tanto una nueva área técnica [44-57] para explorar vías por las que los elementos de la red puedan ser dinámicamente reprogramados por administradores, operadores o usuarios generales para obtener la QoS requerida y otras características, como servicios personalizados. Esto ofrece atractivas ventajas y también cambios importantes en aspectos como el rendimiento, la seguridad y la fiabilidad. Por tanto, ésta es una línea abierta para la investigación y el desarrollo de la ingeniería de protocolos que permita mover el código de servicio (colocado dentro de la capa de transporte de red) a los nodos de conmutación. La bibliografía en este campo estudia varios mecanismos para obtener ventaja de los nodos activos. Algunas de las ventajas de los protocolos activos son conseguidas instalando los nodos activos en puntos estratégicos de la red. Conceptos como redes activas, protocolos *boosters* o agentes software fueron propuestos y desarrollados para redes IP; sin embargo, como hemos dicho antes, las propuestas son insuficientes en el ámbito de las redes de tecnología ATM que es donde hemos situado esta tesis.

Las redes activas proponen otro novedoso paradigma para la construcción de arquitecturas de red basadas en las novedades introducidas por protocolos de comunicaciones como IPv6. Este paradigma se centra en los routers y conmutadores que forman la red, donde los nodos aportan la posibilidad de aplicar acciones de usuarios sobre los paquetes de datos que fluyen por la red. Este planteamiento es parecido a la tecnología basada en agentes móviles, donde los paquetes que fluyen por la red puede ser vistos como pequeños programas que pueden ser ejecutados por los routers y conmutadores. Una distinción [10] que puede destacarse entre los agentes móviles y los pequeños paquetes de código (aunque en muchas investigaciones se suele denominar indistintamente al código móvil y a los agentes móviles) ejecutable es la habilidad de los agentes móviles para retener el estado de un punto de ejecución al siguiente punto.

En la visión integrada de las redes activas los paquetes transmitidos (cápsulas) contienen, además de los datos, fragmentos de programas responsables del procesamiento de los propios datos en los conmutadores. No obstante, también existe un planteamiento discreto en las *active-nets* donde la organización del servicio es realizada separadamente del propio procesamiento del servicio, es decir, lo que conocemos como *out-band*. Se consideran como redes activas las redes de conmutación de paquetes (IP en su gran mayoría) cuyos paquetes pueden contener fragmentos de código para ser ejecutado en los nodos que forman la red. Los nodos extremos de la comunicación se encargan de introducir o inyectar código en la red para cambiar el comportamiento de ésta según sus necesidades. Pero esta inyección de código en la red ha dado lugar a dos corrientes bastante claras en la investigación de las redes activas:

- Conmutadores programables [47,48], pensados para la inyección del código por parte del operador de la red. Ésta es la anteriormente comentada como *out-band*, donde el código es insertado en la red a través de los conmutadores programables preparados al efecto.
- Las cápsulas [49] aportan la visión de que los paquetes de datos transportan pequeños trozos de código de programas que son transportados *in-band* y ejecutados en cada nodo que es encontrado en el camino de la comunicación. Esta visión es la que realmente introduce un nuevo paradigma en las redes de paquetes, que dejan de ser pasivas para ser activas al ser capaces, no sólo de reenviar, enrutar o conmutar paquetes, sino también de ejecutar el código que transportan los paquetes.

Aunque los planteamientos de estas dos ideas son bastante diferentes, las posibilidades de convergencia entre el concepto de red puramente programable y el soporte de cápsulas empiezan a aparecer como factible. Esa posibilidad de coincidencia está en el hecho que hacer programable un nodo requiere más bien de programación especializada que posiblemente el operador de la red no desee realizar y, por otro lado, la generación de cápsulas puede dejarse a la intervención de los operadores de la red. Las cápsulas son más propiamente específicas de las aplicaciones que específicas de los usuarios, por lo que, en realidad pueden desempeñar la función propia del código que los operadores deseen usar al iniciar ciertas aplicaciones.

ANTS (Active Node Transport System) [50] es una de las implementaciones de referencia en el campo de las redes activas. Emplea una variante de la visión *in-band* para construir su arquitectura. En lugar de emplear el transporte constante de código con cada paquete, ANTS almacena el código de usuario más reciente para evitar recargar el código en cada grupo de paquetes que son llamados cápsulas. Las cápsulas

transportan los valores de los parámetros de cada pieza de código. Si el nodo a través del que pasa un paquete contiene el código relacionado, el nodo inicializa el código con los valores de los parámetros de ese paquete y lo ejecuta. Si el código no está presente en ese nodo, el nodo solicita el código a su vecino más cercano en el sentido contrario al del flujo de datos. Hemos de destacar que en cierto sentido hemos inspirado nuestra propuesta en este tipo de transporte de código pero debidamente ajustado a un escenario completamente diferente como es el de la arquitectura TAP.

En realidad, la visión que nosotros damos al protocolo activo está apoyada en las dos vertientes del concepto de redes activas. Por un lado, la existencia de los agentes programables que usamos está relacionada con la ejecución o elección de una CoS concreta. Para nosotros las cápsulas son las células RM que se generan en las retransmisiones que, en alguna de las variantes de nuestra propuesta pueden transportar código muy simple para realizar la búsqueda de las PDUs que hay que retransmitir. Por otro lado, la posibilidad de intervención del operador de la red sobre los agentes programables aporta la vertiente *out-band* en la red activa.

6.11. SISTEMA MULTIAGENTE TAP

Una vez presentados los conceptos principales sobre los que se apoya la tecnología de agentes software, pasamos a describir resumidamente nuestra propuesta, destacando en primer lugar su área de aplicación y clasificación, para comentar después la arquitectura completa del SMA-TAP, así como los agentes que conforman la arquitectura y sus flujos, cuya evolución ha sido propuesta en [58,59].

Comenzamos destacando que la arquitectura TAP está basada en la incorporación de varios agentes software en el modelo de conmutador propuesto y que hemos denominado AcTMs (conmutadores ATM activos). De este modo damos lugar a una red ATM activa en la que sus conmutadores van a desempeñar una función activa en la gestión de las colas de entrada, en la detección de congestiones, así como en la labor de retransmisiones cuando empiezan a perderse células de las fuentes de tráfico que se han elegido con tráfico garantizado.

La *Figura 6.4* permite observar la posición que ocupa el SMA-TAP dentro del modelo arquitectónico ATM. Como puede comprobarse, el SMA forma parte del Plano de Gestión (superficie rayada) ya que aporta nuevos mecanismos para la gestión de recursos ATM, siendo transparente a funciones como CAC y UPC. Pueden observarse también en la *Figura 6.4* las superficies rayadas donde se incorpora nuestra propuesta de protocolo TAP del que forma parte la extensión EAAL-5 que hemos desarrollado.

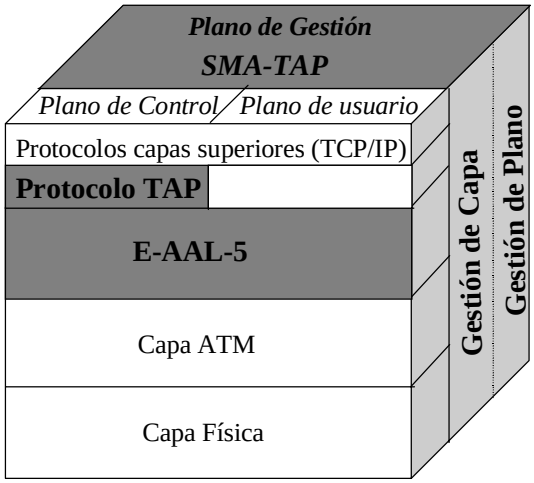

Figura 6.4. Situación del SMA-TAP en el modelo arquitectónico ATM

En el caso de TAP no pretendemos enfrentarnos a escenarios de control de tráfico ni de administración de red, ya que estas tareas ya han sido fuente de múltiples investigaciones, aunque no demasiadas en el ámbito de los SMA. Las funciones CAC y UPC son claras candidatas a ser implementadas mediante agentes software en los que se delegan las labores de establecimiento de la conexión, el control de flujo y la resolución de situaciones de congestión de forma reactiva. Del mismo modo, la mayor parte de investigaciones relacionadas con la aplicación de SMA a ATM están relacionadas con el mantenimiento de los parámetros de QoS, el re-enrutamiento y el tráfico cuando un determinado enlace experimenta problemas.

En la mayor parte de las propuestas, los esfuerzos se han centrado en conseguir el mayor throughput posible. En muchos casos se asume que la QoS es uno de los factores de utilidad más importantes de las redes ATM, pero la maximización del throughput del tráfico en las redes de alta velocidad no es necesariamente el parámetro más valioso para los usuarios de la red, y de hecho, desde el punto de vista de un operador de red ATM, es más importante mantener el servicio dentro de unos márgenes de satisfacción de los usuarios de la red que el de intentar buscar la optimización del ancho de banda arriesgándose a la pérdida de la GoS. Por esto TAP centra sus objetivos en la consecución de la garantía de servicio, aunque es claro que este SMA puede ser fácilmente ampliado con nuevas funcionalidades más propias de la capa de control del modelo de referencia ATM (*Figura 6.4)*.

6.11.1. ARQUITECTURA DEL SMA-TAP

En esta sección presentamos el esquema general de la arquitectura TAP, donde podemos establecer un subsistema constituido por el SMA-TAP que está formado por cinco agentes software.

La *Figura 6.5* presenta la arquitectura TAP completa, incluyendo la memoria DMTE, las colas de entrada, el buffer, las tablas de E/S, así como los agentes software que forman el subsistema SMA-TAP. Las fuentes garantizadas generan su flujo que llega a la DMTE y es procesado a los puertos de salida que multiplexan las células hasta el siguiente conmutador activo.

A continuación comentamos brevemente la funcionalidad de cada uno de los agentes software que serán estudiados detalladamente en el *Capítulo 10*. Los diversos agentes software que constituyen el SMA-TAP son los siguientes:

- El agente CoSA (Class of Service Agent), como podemos observar en la *Figura 6.5*, es un agente programable que se encarga de recibir los parámetros que caracterizan el tráfico que proviene de las fuentes de entrada. En nuestro caso, este agente se encarga del establecimiento de la conexión, de las tablas de routing para determinar los VPI/VCI extremo-extremo y también de la liberación de la conexión.
- El agente WFQA (Weighted Fair Queueing Agent) mantiene una coordinación constante con el agente CoSA, de forma que recibe de éste los flujos de entrada destinados a las colas que WFQA gestiona de forma justa y de acuerdo a los pesos establecidos en los parámetros de tráfico.
- El agente programable CCA (Control Congestion Agent) controla las congestiones basándose en EPDR y en otros algoritmos que el administrador de la red puede elegir. Este agente monitoriza el buffer de entrada, y cuando su ocupación sobrepasa el umbral establecido no acepta ninguna PDU entrante. Del mismo modo, CCA puede ser considerado como un agente inteligente ya que es capaz de autoajustar el valor umbral del buffer en función del tráfico que se está produciendo. Como ya hemos visto en el *Capítulo 5*, la elección del umbral es una labor compleja que está en función del tráfico generado por las fuentes.
- El agente CCA se coordina con el agente RCA para solicitar las PDU al conmutador activo anterior. La función del agente RCA es generar células RM nativas que son transmitidas en sentido contrario al flujo de información. Los conmutadores no activos reconocen las células RM como células RM de TAP y no toman ninguna otra acción que reenviarlas hacia la fuente emisora de los datos.
- El agente DPA se encarga de obtener las PDU o células del buffer de entrada cuando son transmisiones, o PDU de la DMTE cuando se trata de retransmisiones. Una vez recibidas, se encargará de despacharlas completas al correspondiente puerto de salida, previa actualización de la Tabla de Entrada/Salida de ese puerto. Con esta técnica se evitan las congestiones de los puertos de salida y también la mezcla de células pertenecientes a diferentes PDU o fuentes.

6.11.2. CLASIFICACIÓN DEL SMA-TAP

En apartados anteriores se ha realizado la revisión de la literatura en materia de agentes, centrándonos en los aspectos más interesantes relativos a los sistemas de telecomunicaciones. Una vez hecho este ejercicio y, ya presentada la arquitectura general del SMA sobre los conmutadores AcTMs, pasaremos a realizar una clasificación de SMA-TAP atendiendo a la funcionalidad de los agentes que intervienen en el sistema. Antes de ello queremos presentar la forma en que se mapean los agentes en las capas del modelo de referencia ATM. La *Figura 6.6* muestra cómo el agente CoSA está situado en la capa de Control de Plano, mientras los cuatro agentes restantes forman parte de la Capa de Gestión de Planos. Podemos observar también las vías de comunicación entre cada uno de los agentes.

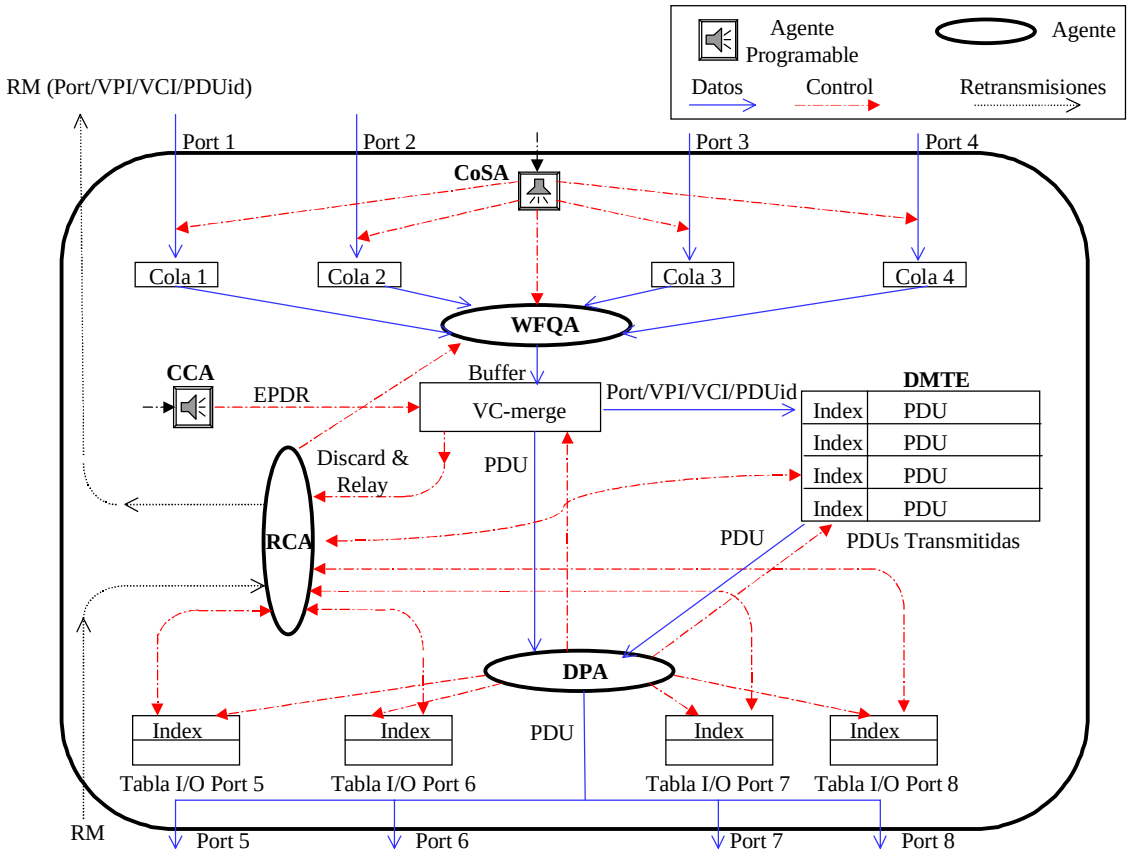

Figura 6.5. *Arquitectura TAP con el subsistema SMA-TAP*

Estamos por tanto ante un sistema multiagente constituido por un grupo de cinco agentes que colaboran conjuntamente para resolver muy diversas tareas. Desde el punto de vista topológico nuestro sistema no se ajusta plenamente a ninguno de los dos planteamientos generales comentados anteriormente. Es decir, no estamos ante un esquema completamente mallado porque no todos los nodos de nuestra red VPN necesitan tener instalado el SMA-TAP para poder disponer de las prestaciones que aporta la arquitectura completa. Por tanto no se requiere la conectividad total de todos los agentes con todos los nodos, lo que reduce sustancialmente, tanto el nivel de complejidad, como la robustez y la escalabilidad del sistema. Tampoco estamos ante una topología jerárquica en el sentido clásico de la palabra, aunque en realidad TAP responde plenamente a este esquema, sólo que de una forma más completa que la presentada en la *Figura 6.4*. Veremos en el Capítulo 7 que estamos ante un sistema claramente distribuido en la red a través de una topología arborescente que soporta las conexiones punto-punto, punto-multipunto y multipunto-multipunto.

Desde el punto de vista funcional de los agentes disponemos de un agente programable, que puede actuar como agente inteligente y también como agente reactivo ante situaciones de congestión. El agente CCA no sólo permite al operador elegir el algoritmo y los parámetros del esquema de control de congestión, si no que se comporta reactivamente ante la aparición de congestiones, y proactivamente pudiendo actuar para evitar la aparición de congestiones ajustando el valor umbral del buffer. Por otro lado, WFQA es un agente colaborativo que recibe eventos desde los agentes CoSA y RCA para llevar a cabo sus acciones generando como salida células y/o PDU clasificadas de acuerdo a los pesos de sus fuentes. El agente CoSA, también programable y reactivo, responde ante el establecimiento de conexión de las fuentes, para lo que necesita coordinarse con otros agentes similares en conmutadores adyacentes. El agente RCA, también reactivo, es capaz de atender eventos provocados por situaciones de congestión en su propio conmutador y en otros conmutadores de la red. Este agente colabora también directamente con el agente WQFA y de forma indirecta con CCA. RCA podría ser considerado como un agente móvil, no en el sentido de los agentes, sino según el punto de vista de las redes activas, porque permite la movilidad de código sencillo entre conmutadores a través de las células RM. Por último, DPA es un agente autónomo que, mediante técnicas de control, se encarga de actualizar las estructuras de datos del sistema como las Tablas de E/S, la memoria DMTE y el buffer del conmutador.

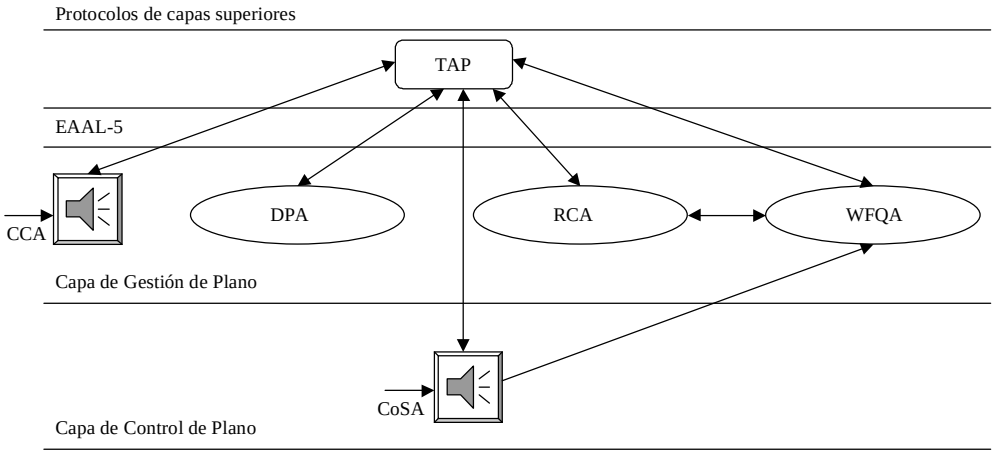

Figura 6.6. *Arquitectura SMA-TAP en capas*

6.12. CONCLUSIONES

Desde luego, no es evidente que la utilización de redes programables, por cualquiera de los métodos comentados, sea claramente útil ya que supone que hay que pagar ciertos costes. Tanto las ventajas como inconvenientes pueden estudiarse desde tres puntos de vista: impacto en los servicios de la red, impacto en el rendimiento de la propia red y facilidad en el desarrollo de protocolos. Existen múltiples investigaciones que demuestran tanto la utilidad de la inclusión de agentes software en la red, como el desarrollo de redes activas, y la mayor parte de ellas acaban beneficiando a uno o todos los puntos de vista anteriores. Este capítulo ha aclarado conceptos y presentado una taxonomía del ámbito de las redes programables, donde podemos comprobar la amplitud de horizontes planteados en las investigaciones actuales. Estas investigaciones indagan en la introducción e integración de agentes programables y agentes móviles en las arquitecturas emergentes de telecomunicaciones. El concepto general de agentes aporta un nuevo paradigma por la rápida y parametrizable provisión de servicios en entornos de procesamiento distribuido. Las investigaciones presentadas proponen mejoras para solventar ciertas limitaciones de las propuestas estándares y aprovechamos para incluir en la tecnología ATM un importante objetivo final como es la visión de redes ATM activas o redes programables abiertas (open programmable networks).

Uniendo la visión de los SMA y de las redes activas hemos justificado la propuesta de nuestro SMA-TAP que se basa en una arquitectura de cinco agentes software que soportan el protocolo TAP sobre los conmutadores activos AcTMs inspirados en algunos de los fundamentos de las redes activas. De este modo nos encontramos ante un protocolo activo capaz de aportar la GoS deseada a las conexiones que el usuario o el operador de la red decidan. El protocolo TAP que proponemos es activo desde los dos puntos de vista de las redes activas: porque la arquitectura que lo soporta dispone de nodos activos en puntos estratégicos de la red que implementan un protocolo activo que permite que el código de usuario sea cargado en los nodos de la red en tiempo de ejecución, y porque, además, la arquitectura y el protocolo soportarían la propagación de código gracias al uso de células RM que se pueden encargar de transportar tanto código como datos.

REFERENCIAS

[1] *Diccionario de la Lengua Española. Real Academia Española. XXI Edición*, (1992).

[2] *Webster's Encyclopedic Unabridged Dictionary of the English Languaje*. Ed. Gramercy (1996).

[3] Object Management Group, "Agent Technology Green paper," *OMG Document ec/99-12-02*, (1999).

[4] S. Franklin, A. Graesser. "Is it an Agent or just a Program?: A Taxonomy for Autonomous Agents," *Third International Workshop on Agent Theories, Architectures, and Languages*, *Springer-Verlag*, (1996).

[5] Wooldridge, M. and Jennings, N. R., "Intelligent Agents: Theory and Practice," *The Knowledge Engineering Review. Vol. 10. No. 2*, (1995).

[6] Alex L. G. Hayzelden and John Bigham (Eds.), "Software Agents for Future Communications Systems," *Springer*, (1999).

[7] David D. Clark and David L. Tennenhouse, "Architertural considerations for a new generation of protocols," *ACM SIGCOMM*, (1990).

[8] Nwana, H. S., Azarmi, N. (Eds.) "Sofware agents and Soft Computing: Towards Enhancing Machine Intelligence," *Springer-Verlag*, (1997).

[9] Müller, J. P., "Architecture and Applications of Intelligent Agents: A survey," *The Knowledge Engineering Review, Vol. 13. No. 4*, (1998).

[10] Alex L. G. Hayzelden and John Bigham, "Agent Technology in Communications Systems: An Overview," *http://www.agentcom.org/agentcom/*, (1999).

[11] Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., Evans, R. "Software Agents: A review," *Technical Report of Trinity College. University of Dublin. http://www.cs.tcd.ie/research_groups/aig/aig/*, (1997).

[12] Magedanz, T., Rothernel, K., Krause, S., "Intelligent Agents: An emerging Technology Next Generation Telecommunications?," *INFOCOM'96*, (1996).

[13] Russell, S. Norvig, P., "Artificial Intelligence: A modern Approach," *Prentice Hall*, (1995).

[14] H. S. Nwana. "Software Agents: An Overview," *Knowledge Engineering Review*, (1996).

[15] N. R. Jennings, K. Sycara, M. Wooldridge. "A Roadmap of Agent Research and Development," *Autonomous Agents and Multi-Agent Systems*, I, (1998).

[16] M. P. Singh. "Agent Communication Languages: Rethinking the Principles," *IEEE Computer Vol. 31*, (1998).

[17] Pere Vilà and Josep L. Marzo, "Scalability Study and Distributed Simulations of an ATM Network Management System based on Intelligent Agents," SPECTS'2K 2000 SCS Symposium on Performance Evaluation, (2000).

[18] L.C. Lee, H. S. Nwana, D. T. Ndumu and P. De Wilde, "The stability, scalability and performance of multi-agent systems," *http://www.labs.bt.com/projects/agents/publish/papers.htm*, (1998).

[19] Mark Greaves, Heather Holmback and Jeffrey Bradshaw, "What Is a Conversation Policy?," *KAW'99 Proc. of the Workshop on specifying and implementing conversation*, (1999).

[20] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," *Software Agents, The AAAI Press* pp. 291-316, (1997).

[21] D. Steinerark, "FIPA 97 Specification Version 2, Part 2: Agent Communications," *Foundation of Intelligent Physical Agents, http://www.fipa.org/spec/FIPA97.html*, (1997).

[22] ITU-T Rec. M .3010 Principles for a Telecommunications Management Network (TMN), *Study Group IV*, (1996).

[23] TINA Consortium, Overall Concepts and Principles of TINA, *Document label TB_MDC.018 1.0_94, TINA-C*, (1995).

[24] David Grifin, George Pavlou and Panos Georgatsos, "Providing Customisable Network Management Services Through Mobile Agentes," *http://www.slpl.lib.mo.us/cco/minutes/103098.htm*, (1998).

[25] Object Management Group, The Common Object Request Broker: Architecture and Specification (CORBA), Versión 2.0, (1995).

[26] Mobile Agent System Interoperability Facilities Specification OMG TC Document orbos/97-10-05, ftp://ftp.omg.org/pub/docs/orbos/97-10-05.pdf, (1997).

[27] Somers, Fergal, "Hybrid:unifying centralised and distributed network management using intelligent agents," *IEEE Network Operations and Management Symposium NOMS'96*, pp. 34-43 vol. 1, (1996).

[28] Jim Hardwicke and Rob Davison, "Software Agents for ATM Performance Management," *Networks Operations and Management Simposium NOMS 98, IEEE*, Volume 2, pp. 313-321, (1998).

[29] Somers, F. "HYBRID: Intelligents Agents for Distributed ATM Network Management," *IATA Workshop at ECAI'96*, (1996).

[30] Frei, C. and Faltings, B. "Intelligent Agents for Network Management," *AI for Network Management Systems, IEE Digest No. 97/094*, (1997).

[31] Gaïti, D. And Boukhatem, N. "Cooperative Congestion Control Schemes in ATM Networks," *IEEE Communications Magazine*, Vol. 34 No. 1, (1996).

[32] MASIF-RTF Results. Object Management Group, (1998).

[33] Pavón, J., Tomás, J. Bardout, Y., Hauw, L. H. " CORBA for Network and Service Management in the TINA Framework," *IEEE Communications Magazine*, Vol.36, No 3, (1996).

[34] W. H. Andrag, and C. W. Omlin, "Distributed Intelligent Multi-Agents for Telecommunication Network Management," *http://www.cs.sun.ac.za/projects/tech_reports/*

[35] Simon Steward and Steve Appleby, "Mobile software Agents for Control of Distributed Systems Based on Principles of Social Insect Behaviour," *Proceedings Conference Singapore ICCS'94* pp. 549-553, (1994).

[36] Gwan Joong Kim, Young Sun Kim and Hyeong Ho Lee, " A design of Management System for ATM Switches Using Mobile Agent Concept," *Global Telecommunications Conference, GLOBECOM 1998. The Bridge to Global Integration. IEEE*, pp. 1694-1698 vol.3. (1998).

[37] J. Gregoire, "Models and Support Mechanisms for Distributed Management," *Proceedings of the 4th International Symposium on Integrated Network Management*, (1995).

[38] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini, "Decentralizing Control and Intelligence in Network Management," *Proceedings of the 4th International Symposium on Integrated Network Management*, (1995).

[39] M. Suzuki, Y. Kiriha, S. Nakai, "Dynamic Script Binding for Delegation Agent," *Proceedings of the 7th IFIP/IEEE International Workshop on Distributed Systems Operation & Management*, (1996).

[40] T. Magedanz, T. Eckardt, "Mobile Software Agents: A new Paradigm for Telecommunications Management," *Proceedings of IEEE/IFIP 1996 Network Operations and Management Symposium*, (1996).

[41] Y. Yemini, G. Goldszmidt and S. Yemini, "Network Management by Delegation," *Proceedings of the 2nd International Symposium on Integrated Network Management*, (1991).

[42] G. Cugola, C. Ghezzi, G. P. Picco, and G. Vigna, "A Characterization of Mobility and State Distribution in Mobile Code Languajes," *Proceedings of the 2nd Workshop on Mobile Objects Systems*, (1996).

[43] Jiann-Liang Chen, Shingfeng Lin, Ming-Yi Lee, "Distributed fault management over ATM networks," *Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp.196-201, (1997).

[44] Tennenhouse, D. Smith, J., Sincoskie, W.D. and Wetherall, D. "A Survey of Active Network Research," *IEEE Communications Magazine*, pp. 80-85, (1997).

[45] David L. Tennenhouse, and David Wetherall, "Towards an Active Network Architecture," *Multimedia Computing and Networking*, (1996).

[46] Dan S. Decasper, Bernhard Plattner, Guru M. Parulkar, Sumi Choi, John D. DeHart, and Tilman Wolf, "A scalable high performance Active Network Node," *IEEE Network*, pp. 8-19, (1999).

[47] D. Alexander, "Active bridging," *Proc. SIGCOMM'97*, (1997).

[48] S. Bhattacharjee, "An architecture for active networking," *Proc. INFOCOM'97*, (1997).

[49] D. Wetherall and D. Tennenhouse, "The ACTIVE IP Options," *Proc. ACM SIGOPS*, (1996).

[50] David Wetherall, John Guttag and David Tennenhouse, "ANTS: A Toolkit for building and dynamically deplying network protocols," *IEEE OPENARCH'98*, (1998).

[51] Ulana Legedza, David Wetherall and John Guttag, "Improving the performance of distributed applications using Active Networks," *INFOCOM'98*, pp. 590-599, (1998).

[52] D.A. Halls and S. G. Rooney, "Controlling the Tempest: Adaptive Management in Advanced ATM Control Architecture," *IEEE JSAC*, Vol. 16, Nº 3, pp. 414-423, (1998).

[53] G. Pujolle and D. Gaïti, "ATM Flow Control Schemes Through a Multi-Agent System," *International Conference on Information Engineering'93, Proceedings of IEEE*, Volume 1, pp. 455-459, (1993).

[54] Jiann-Liang Chen, Ying-Ping Yu, and S. Lin, "Exploiting Multi-Agent Scheme for Traffic Management in ATM Networks," *Intelligent Control (ISIC), Proceedings, IEEE*, pp. 594-599, (1998).

[55] German Goldszmidt, and Yechiam Yemini, "Delegated Agents for Network Management," *IEEE Communications Magazine*, Volume 36, 3, pp. 66-70, March (1998).

[56] G. Goldszmidt, and Y. Yemini, "Distributed Management by Delegation," *IEEE 15th International Conference on Distributed Computing*, pp. 333-340, (1995).

[57] F. Nait-Abdesselam, N. Agoulmine, and A. Kasiolas, "Agents Based Approach for QoS Adaptation In Distributed Multimedia Applications over ATM Networks," *IEEE International Conference on ATM, ACATM-98*, pp. 319-326, (1998).

[58] José Luis González-Sánchez and Jordi Domingo-Pascual, "TAP: Architecture for Trusted Transfers in ATM Networks with Active Switches," *ATM'2000 IEEE Conference on High Performance Switching and Routing Joint IEEE ATM Workshop 2000 and 3rd International Conference on ATM (ICATM'2000)*, pp. 105-112 (2000).

[59] José Luis González-Sánchez and Jordi Domingo-Pascual, "Trusted and Active Protocol over a Distributed Architecture in ATM Networks with agents," *IEEE International Conference on Computer Communication and Networks (IEEE IC3N'2000)*, pp. 484-490 (2000).

Direcciones Universal Resource Locator (URL).

[URL1] Craig Thompson C. and Odell J., Agent Technology Glossary "OMG Agent WG Technology Green Paper", *Object Management Group: http://www.objs.com/agility/tech-reports/9909-agent-glossary.html* .

[URL2] Grasshopper: The Agent Platform (IKV++): *http://www.ikv.de/products/grasshopper/*

[URL3] TINA-C *http://www.tinac.com*

[URL4] (Ramón M. Gómez Labrador, Universidad de Sevilla), *http://www.fie.us.es/~ramon/*

[URL5] KQML (AgentWeb, Universidad de Maryland en Baltimore): *http://www.cs.umbc.edu/kqml/*

[URL6] JATLite (CDR, Universitdad de Stanford). *http://java.stanford.edu/java_agent/html/*

[URL7] FIPA Specifications: *http:// www.fipa.com*

[URL8] France Télécom: *http://www.francetelecom.fr/*

[URL9] Cluster CLIMATE ACTS *http://www.fokus.gmd.de/research/cc/ima/acts-d5-ac/*

[URL10] Odyssey y Telescript (General Magic): *http://www.genmagic.com/technology/odyssey.html*

[URL11] Java Technology Home Page (Sun Microsystems): *http://java.sun.com/*

[URL12] The Tcl Company (Scriptics): *http://www.scriptics.com/*

[URL13] Proyecto Tele-MACS *http://www.agentcom.org/agentcom/*

[URL14] Agentes Inteligentes *http://bogart.sip.ucm.es/~juan/agentes/am.pdf*

[URL15] Estándares OMG *http://www.omg.org/*

[URL16] Active Networks home page of MIT *http://www.tns.lcs.mit.edu/activeware/*

[URL17] Proyecto ACTS IMPACT *http://www.acts-impact.org/impact/*

[URL18] The Agent Society: *http://www.agent.org/*

[URL19] The Aglets Software Development Kit (I.B.M. Corp.): *http://www.trl.ibm.co.jp/aglets/*

[URL20] Mobile Objects and Agents (Open Group): *http://www.caab.opengroup.org/RI/java/moa/index.htm*

[URL21] Telemedia, Networks and Systems Group *http://www.tns.lcs.mit.edu/*

[URL22] Agentes y Sistemas Multiagente *http://Montealegre.ei.uvigo.es/Agentes/*

[URL23] *http://www.cetus-links.org/oo_mobile_agents.html*

CAPÍTULO 7

TAP COMO SISTEMA DISTRIBUIDO

7.1. INTRODUCCIÓN

Las posibilidades de la tecnología ATM con anchos de banda escalables entre los 155 Mbps y varios Gbps le aportan grandes posibilidades para su diseño como sistema distribuido. Sin embargo, algunas de las cuestiones que ya hemos comentado, como la orientación a la conexión de ATM, el control de flujo y de errores fuera de la red, y las retransmisiones extremo-extremo le dan una apariencia muy alejada de la visión clásica de los sistemas distribuidos. Sabemos que los conmutadores presentes en la red pueden descartar células cuando aparecen las congestiones. Sin embargo, cuando estas células forman parte de PDU, la pérdida de una célula es detectada en el destinatario, lo que implica que deberá realizarse la solicitud de retransmisión desde el origen, previa solicitud por parte del destinatario. Este escenario nos sitúa por tanto en una posición en la cual se justifica la necesidad de mecanismos distribuidos en la red que permitan poder solventar los problemas de forma local a donde aparecen. De este modo, la responsabilidad de resolver problemáticas como la congestión, puede ser distribuida a lo largo de toda la red sin necesidad de implicar a los extremos de la conexión, ni tampoco a aquellos tramos de la red que no experimentan congestiones. Precisamente, lo que nos proponemos con TAP es la posibilidad de distribuir los conmutadores que soportan el protocolo de GoS a lo largo de una red VPN, lo que aporta a la tecnología ATM esa característica de sistema distribuido que, en no pocas ocasiones, se le ha reclamado.

Podemos adecuar la definición realizada en [1] sobre lo que es un SD (Sistema Distribuido) y ajustarla a nuestras necesidades concretas, de modo que podemos decir que un SD consiste en un conjunto de dispositivos de cómputo (ordenadores, conmutadores y elementos de interconexión) unidos todos por una red y equipados con un sistema software no centralizado. De este modo, el software no centralizado conduce a la idea de un sistema software distribuido que permite a los elementos hardware del SD coordinar sus actividades y compartir los recursos del sistema (hardware, software y datos). El objetivo final del SD es conseguir que sus usuarios lo vean como un único e integrado entorno de computación, aunque en realidad se trate de diferentes elementos dispersos geográficamente e interconectados por una red.

Los agentes software distribuidos (multiagentes) son una atractiva y emergente tecnología que permite el procesamiento de información distribuida. Los objetivos de estos agentes son los dos principios de la construcción de sistemas complejos: la construcción de agentes múltiples y de mecanismos para la coordinación de agentes de comportamiento independientes. Los agentes son útiles en aplicaciones donde el entorno cambia dinámicamente y su comportamiento inteligente es deseable y hasta esencial (como en la toma de decisiones, predicción y planificación). En estas aplicaciones , estas tareas son realizadas por un conjunto de programas en paralelo pero independientes. Incluso, los entornos en los cuales, por su naturaleza no requieren de sistemas distribuidos, pueden beneficiarse del uso de sistemas multiagente. Estos beneficios son la escalabilidad, robustez y programación más sencilla. En el caso de las comunicaciones, una de estas aplicaciones es la gestión de la red, pero otras pueden ser el routing, monitorización del rendimiento, gestión de conexiones, control de congestiones, etc.

Tradicionalmente, la gestión de los sistemas de telecomunicaciones se ha realizado a través de sistemas centralizados, muy seguros, pero con importantes carencias como la dependencia de funcionamiento de todo

el sistema de elementos concretos de la red, como la escasa escalabilidad y la elevada complejidad, por citar algunas de ellas. La aparición de tecnologías como ATM, que permite la reconfiguración dinámica de los recursos de la red en tiempo de ejecución, ha dado lugar a nuevos servicios que ya han sido comentados en capítulos previos. No obstante, aunque la tecnología ATM ha sido concebida para aportar estos y otros requerimientos, aparece un importante problema como es el resolver la complejidad resultante de estas demandas. Una forma de resolver este problema es hacerlo mediante un control jerárquico de la red, o bien distribuyendo el control de la red entre agentes software que se encargan de gestionar tareas concretas. Precisamente, y como hemos explicado en el *Capítulo 6*, nuestra propuesta va en esta línea.

Algunas de las ventajas de los SMA son la escalabilidad y la tolerancia a fallos. Esto se deriva directamente del carácter distribuido, de la gestión adaptativa del tráfico realizado por los agentes. Como las redes suelen crecer, tanto en tamaño como en complejidad, el control distribuido y robusto pasa a jugar un papel primordial que es más importante cada día. Los cambios en la topología de la red pueden ser manipulados incorporando más agentes a la red. De este modo, cuando un nodo falla, como es nuestro caso en las congestiones, otro agente puede aprender a tomar el control para cambiar y reenrutar el tráfico nuevamente, o actuando activamente en la recuperación del tráfico hasta que el conmutador congestionado vuelve a su estado de normalidad.

En nuestro caso, la visión de sistema distribuido nos la aporta el SMA que empleamos para construir la arquitectura propuesta y también el propio protocolo TAP que engloba al SMA. Como en los SD [1], en el caso de los SMA podemos hablar de tiempo de respuesta, rendimiento, usuarios concurrentes o tareas paralelas. Y desde este punto de vista vamos a revisar algunas de esas propiedades de los SMA que tienen cierta relación con el concepto de SD. TAP es una arquitectura basada en un esquema de sistema multiagente y, como tal, podemos decir que se trata de una arquitectura distribuida en la topología de la VPN que la incorpora. De forma general TAP puede formar parte de un sistema de gestión de red[1] que aporte a la tecnología ATM todas las características de las que ésta carece y que han sido descritas en capítulos precedentes.

7.2. SISTEMAS Y ARQUITECTURAS DISTRIBUIDAS

Tradicionalmente se ha diferenciado entre cuatro niveles de arquitecturas para el control de problemas complejos: centralizados, jerárquicos, distribuidos e híbridos. Cada una de ellas tiene sus ventajas e inconvenientes, pero en el caso de las redes de telecomunicaciones podemos decir que la tendencia es que su control es claramente distribuido. Son muchas las definiciones de SD que pueden encontrarse en la literatura, y una de ellas en [2] indica que es una colección de computadores independientes que aportan a sus usuarios la sensación de trabajar en un único ordenador. Quizás la mejor forma de entender el concepto de SD es contrastarlo con los sistemas centralizados donde se depende de un elemento central que controla todo el sistema. Así, algunas de las ventajas de los SD respecto a los centralizados son: la velocidad, la fiabilidad, el crecimiento incremental, la distribución de las aplicaciones y los aspectos económicos. Otras propiedades importantes son: la compartición de datos, de servicios, las ventajas de comunicación entre diferentes componentes de un entorno y también la escalabilidad y la robustez ante posibles fallos del sistema. En cualquier caso, la resolución de problemas de forma distribuida aporta una gran potencialidad que permite incrementar la capacidad de cómputo y la disponibilidad de recursos de redes en aplicaciones locales y/o de mayor alcance.

Existen algunas cuestiones básicas para el diseño [1,2] y gestión [3] de SD y, entre ellas, destacamos las siguientes, que podemos ver como estrechamente relacionadas con algunos de los parámetros de QoS que ATM es capaz de garantizar:

- Transparencia: desde el punto de vista que los usuarios del SD puedan disponer de sus recursos, como si se encontrasen ante un sistema centralizado. Es decir, mientras se están disfrutando todas las ventajas de un sistema distribuido se tiene la impresión de disponer de las ventajas de un sistema centralizado. De este modo la característica de transparencia puede ser entendida desde diversos puntos de vista. Por un lado se habla de transparencia en cuanto a la localización física de los recursos que se usan. Transparencia en cuanto a la migración de los recursos, es decir, podemos mover geográficamente los recursos del SD sin afectarlo y manteniendo sus nombres. Transparencia en cuanto a la replicación de los recursos, que permite que el usuario del SD no necesite conocer cuántos recursos están replicados para aportar tolerancia a fallos al sistema. Transparencia respecto a la

[1] Tradicionalmente, se considera a los sistemas de gestión de red como una serie de herramientas para la monitorización y control de la red que suelen integrarse en atractivos entornos de usuario que permite la ejecución de comandos pensados para facilitar las labores de administración de la red.

concurrencia que permite que múltiples usuarios del SD puedan compartir conjuntamente sus recursos sin conocer este hecho. Por último, transparencia respecto al paralelismo que permite que las tareas realizadas en el SD se realicen en paralelo sin tenerse conocimiento de ello.

- Flexibilidad: En el caso de los Sistemas Operativos Distribuidos existen dos grandes corrientes en cuanto al diseño del SD. Por una lado está la corriente a favor de los sistemas monolíticos con un núcleo (kernel) que aporta la mayoría de los recursos, y por otro, la corriente partidaria de sistemas de microkernel que distribuyen las tareas y recursos entre varios equipos que disponen del menor número posible de recursos, pero sin perder de vista que todo el SD completo debe ofrecer todos los recursos de los sistemas separados. La posibilidad de elegir una corriente u otra, o los grados intermedios entre ambos extremos, es lo que podemos denominar como flexibilidad del SD.
- Fiabilidad: Este ha sido uno de los objetivos más importantes de los SD desde sus orígenes, y en nuestro caso particular es una de las aspiraciones principales. El planteamiento general justifica el uso de SD porque al distribuir los recursos y la carga, la fiabilidad del mismo es mucho superior que si se dispone de un sistema centralizado que se encargue de soportar todos los recursos y la carga. En el sistema centralizado cuando falla el nodo que lo soporta, todo deja de funcionar, mientras que si se trata de un SD la probabilidad de fallo se reparte entre los componentes del mismo.
- Funcionamiento: el rendimiento del SD debería ser superior al de un sistema centralizado para que su uso tenga justificación. En cierto modo, es cuestión de buscar el punto de equilibrio entre el rendimiento final del sistema y el resto de características comentadas. En algunos casos podría tener sentido sacrificar el rendimiento por la fiabilidad pero, en general, lo interesante es ser capaz de garantizar todas las características, entre las que el funcionamiento eficiente puede ser de las más importantes. En cuanto a la forma de medir el rendimiento pueden usarse varias técnicas como el tiempo de respuesta ante ciertos eventos, el índice de utilización del sistema, el número de accesos garantizados, el ancho de banda consumida en una red, el throughput final del sistema, etc. En nuestro caso una de las mayores aspiraciones está en conseguir una optimización del goodput de la red.
- Escalabilidad: esta propiedad estudiada en [4] debe permitir que el SD soporte un numero variable (generalmente elevado) de usuarios sin degradar el índice de rendimiento visto anteriormente. Esta es una característica que depende, tanto de aspectos software como hardware. En muchos casos, un recurso hardware (por ejemplo, un disco) puede acabar degradando el rendimiento final del sistema cuando el índice de simultaneidad de accesos de usuarios al disco crece por encima de un determinado valor. Sin embargo, también puede producirse el mismo problema de escalabilidad en aspectos software como podría ser un algoritmo o protocolo centralizado que atiende todas las peticiones de los usuarios que acceden al sistema.

Pero también aparecen desventajas en la utilización de SD como, por ejemplo, pérdida de seguridad y posibles problemas en la red. El diseño de SD debe conducir a características como la transparencia, flexibilidad pero, sobre todo, son atractivos para nosotros la fiabilidad, el rendimiento y la escalabilidad.

El carácter distribuido de los sistemas es posible gracias a la existencia de una red que permite la interconexión de todos los recursos del mismo. Por tanto, necesitamos nuevamente de elementos hardware y software para implementar la red que hace de nexo en el SD. Por un lado es necesaria la existencia de cableado y elementos de interconexión a la red y, por otro lado, es imprescindible la implementación de un protocolo de comunicación que permita, como poco: la especificación de la secuencia de mensajes que se van a intercambiar con la red y la especificación del formato de los datos de los mensajes intercambiados. Sabemos que un protocolo es implementado en un par de módulos software localizados en los dos (o más) extremos de una comunicación. Si además, los elementos de interconexión de la red que une los extremos de la comunicación implementan el protocolo podemos decir que éste está distribuido en la red. En el caso de ATM sabemos que se trata de una tecnología de conmutación de paquetes basada en un mecanismo de enrutamiento de paquetes conocida como *cell relay* que opera a mucha mayor velocidad que la conmutación de paquetes tradicional. Precisamente, se intenta conseguir esa velocidad de transmisión evitando que los nodos intermedios de la red se encarguen de las labores de control de flujo y de control de errores. Esto unido al carácter orientado a la conexión nos hace perder de vista el carácter distribuido de ATM. TAP aporta ese carácter distribuido a la red como un protocolo implementado en los extremos de la comunicación y en los conmutadores activos. Debemos destacar que la inclusión de TAP no resta rendimiento a la red, ya que el protocolo lo que hace es aprovechar los instantes de silencio de las fuentes para realizar las retransmisiones que sean solicitadas por las conexiones con requerimientos de GoS.

7.3. PROPIEDADES DE FUNCIONAMIENTO DE LOS SMA DISTRIBUIDOS

Según hemos visto en el *Capítulo 6*, un SMA está constituido por un grupo de agentes autónomos que interactuan unos con otros para alcanzar un objetivo común o individual. Para ello, los agentes deben coordinar sus acciones, para dar respuesta o soluciones como: asignar recursos; evitar o eliminar conflictos en la ejecución de acciones o por conflicto de intereses; conseguir eficiencia mejorando la predicción o reduciendo la redundancia. En líneas generales, lograr una eficiencia global del entorno en que se desarrollan que, en nuestro caso, es el ámbito de los sistemas de telecomunicaciones.

Pero en los SMA no sólo tienen importancia las características puramente funcionales[2] como la modelización del conocimiento, la movilidad, la inteligencia, la coordinación, etc. Debe prestarse también atención a otras propiedades no funcionales de los SMA, que son las relativas al funcionamiento[3] del propio SMA, como la escalabiliad, el rendimiento y la estabilidad. Estas características van recibiendo cada día mayor atención a medida que los SMA van madurando e implantándose [5]. Por esto vamos a centrar estos tres importantes conceptos.

- Tanto el rendimiento[4] o funcionamiento, como el throughput, el tiempo de respuesta y el número de usuarios concurrentes tienen también un papel importante en los SMA. Pero existen otros factores importantes que tienen costos y, por tanto, afectan al rendimiento. Podemos destacar los siguientes:
  - ✓ Coordinación de los protocolos usados, que suelen conllevar razonamiento y comunicación que llevan asociados costes computacional y de interconexión.
  - ✓ Modelo de conocimiento de los agentes que requiere de estructuras de datos complejas con costes de almacenamiento y de manipulación.
  - ✓ Modelo de racionalidad de los agentes que conllevan también un coste computacional debido a la evaluación y optimización del razonamiento.

  Por tanto, lo mismo que en los sistemas distribuidos, los indicadores de rendimiento en los SMA se expresan como costes computacionales y en throughput. El rendimiento es una función de la complejidad computacional de la implementación del SMA. En [5] se propone una definición de rendimiento en los SMA como medida en la que se usa un conjunto de indicadores estadísticos de las salidas del sistema y sus consumos de recursos donde algunos de ellos son el throughput, el tiempo de respuesta, el número de agentes o tareas concurrentes, el tiempo computacional y las sobrecargas debidas a las labores de comunicación.
- En cuanto a la escalabilidad, en el caso de los SMA, se refiere al modo en que incrementa la eficiencia de la función del sistema a medida que aumenta la complejidad de éste. Así, en la ingeniería del software distribuido el término escalabilidad se suele usar para referirse al incremento de la carga del entorno debido a un incremento en el número de componentes distribuidos. En el caso de los SMA el incremento en la carga del sistema es causada por la necesidad de mayor interacción a medida que crece el número de agentes que forman parte del SMA. La escalabilidad es por tanto una medida media de la degradación del rendimiento de los agentes individuales del sistema causada por el crecimiento del tamaño del sistema. Es decir, estamos ante un visión de complejidad algorítmica, ya que la escalabilidad del SMA depende, en el peor de los casos, del rendimiento del sistema que está limitado por una función polinómica de la carga del mismo.
- La estabilidad es una propiedad del equilibrio. Es decir, un sistema es estable si tras ser perturbado es capaz de volver voluntariamente a la situación de equilibrio original. En el caso de los SMA la estabilidad suele definirse en torno a las perturbaciones como el ruido, variación de los valores de ciertos parámetros del sistema, o las congestiones como es nuestro caso. El sistema será considerado inestable si después de un tiempo es incapaz de volver a la situación de equilibrio. En el caso de los SMA el punto de equilibro es difícil de definir, aunque se dice que están en equilibrio cuando las propiedades estadísticas de sus indicadores de rendimiento permanecen estacionarias para una variación dada en la carga externa del sistema. En nuestra arquitectura la situación de inestabilidad se produce cuando aparecen las congestiones en los conmutadores.

---

[2] Consideramos como propiedades funcionales aquellas que están relacionadas con la actividad vital y particular de cada agente individualmente o con el resto. Es decir, las funciones desempeñadas por cada agente del sistema.

[3] Definimos las propiedades de funcionamiento como aquellas que reflejan el grado de funcionamiento del SMA completo, que lógicamente viene determinado por el funcionamiento individual de cada uno de los agentes, o por la forma en que cada uno desempeña sus funciones.

[4] En el caso de los SMA el throughput suele expresarse como el número de transacciones por unidad de tiempo que el sistema es capaz de procesar.

Los sistemas expertos distribuidos son considerados como una de las grandes áreas de aplicación del procesamiento distribuido. Esta área es una de las que más nos interesa en nuestro caso, ya que las aplicaciones de sistemas expertos distribuidos destacan cuántos agentes negocian sobre soluciones colectivas. Uno de los objetivos de diseño de las aplicaciones distribuidas y de tiempo real ha sido el de reducir la frecuencia de comunicación entre agentes, o elevar la velocidad de comunicación de los recursos del SD. Sin embargo, las características de cada aplicación son las que definen el número de agentes que se necesitan, por lo que se requiere un cierto grado de comunicación entre los agentes. La literatura describe también paradigmas distribuidos y la referencia [6] presenta este paradigma diseñado por un sistema multiagente distribuido, donde cada agente en su entorno es capaz de computar autónomamente y cooperar con otros agentes para buscar una solución a un problema concreto.

Por otro lado, los agentes móviles también aportan atractivas características al procesamiento distribuido en un intento por evitar las debilidades de los sistemas centralizados como la falta de escalabilidad del control central, el incremento en los procesos y comunicaciones y su vulnerabilidad a los fallos en el sistema. Los procesos distribuidos necesitan comunicar entre sí para identificar y detectar eventos. De esta forma los agentes pueden aportar nuevas formas de enfrentarse a la problemática del control de los sistemas distribuidos para reducir su complejidad. Las referencias [7, 8] proponen la aplicación de los agentes móviles en el control de los SD, aunque podemos destacar que en realidad, la mayor parte de las investigaciones relativas a la aplicación de la tecnología de agentes a los sistemas de comunicaciones acaban reconociendo el carácter inherentemente distribuido de los resultados obtenidos.

7.4. PROTOCOLO TAP SOBRE UNA VPN DISTRIBUIDA

El concepto formal de VPN (Virtual Private Network) tiene una cierta similitud con las redes auto-contenidas, de forma que la VPN, además de ser virtual, supone que no está conectada a redes externas. Se supone que los datos son intercambiados de forma secreta entre los nodos. Además se supone que no hay conocimiento externo de la existencia de la red virtual. Cisco Systems [10] presenta su concepto de VPN indicando que se trata de "*un entorno de comunicaciones en el cual el acceso es controlado para permitir conexiones sólo dentro de una comunidad de interés, que es construido sobre alguna forma de partición de un medio de comunicación subordinado, medio que aporta servicios a la VPN de una forma no exclusiva*".

Existen diferentes definiciones de VPN para diferentes problemas y soluciones diferentes, por lo que nosotros deseamos relajar en cierto modo los anteriores conceptos de VPN y queremos quedarnos con la calificación de "discreta" en sus acepciones de separada, distinta o disjunta. No entramos por tanto en las características de independencia de redes exteriores, pero sí deseamos aclarar que nuestra propuesta de emplear TAP sobre una VPN pretende indicar que estamos ante una red privada o corporativa de tecnología ATM y que en su topología cuenta con conmutadores AcTMs que soportan la arquitectura y el protocolo TAP. De este modo, podemos considerar una VPN como una red privada construida dentro de una infraestructura de red pública, como sería en nuestro caso una red LAN, MAN o WAN de tecnología ATM.

La propuesta de esta VPN para el soporte de TAP coincide con las motivaciones generales de las VPN que se basan en criterios económicos, de privacidad, seguridad y, sobre todo, en aspectos de QoS. Al margen, hemos de destacar que, dadas las características comentadas hasta ahora y la justificación de SD que realizaremos en las siguientes secciones, justifican que TAP podría ser también usado en conexiones con requerimientos de GoS a través de troncales ATM. Quizás nuestra motivación más importante para proponer esta VPN es virtualizar o hacer invisible una parte de la red a usuarios externos de la VPN, mientras los usuarios internos pueden disfrutar de las ventajas que aporta TAP. No obstante, para enmarcar adecuadamente nuestros escenarios de trabajo nos restringimos a lo que denominamos VPN distribuida.

Mientras las velocidades de transmisión aumentan, los ACK acaban teniendo costes casi constantes entre los extremos por el tiempo *RTT* (R*ound-Trip Time*). Es decir, el ancho de banda en la red tiende a cero a medida que las fuentes incrementan sus requerimientos, pero el *RTT* se acerca asintóticamente a valores constantes que dependen del medio de transmisión que usemos. Las congestiones provocan el problema de los retardos, pero además, el control de flujo para acondicionar las prestaciones de emisor, receptor y red conduce también a desaprovechar los recursos de la red. Esto ocurre porque el ajuste provoca desaprovechar en muchos casos el ancho de banda disponible en la red que lleva hasta la posibilidad de que ésta quede desaprovechada estando inactiva en márgenes elevados del ancho de banda. Estas apreciaciones forman parte de las motivaciones generales de nuestra investigación, las cuáles van a ser descritas en *Capítulo 8*. Lo que podemos apreciar, como poco, es que las latencias debidas a los tiempos de *RTT* podrían ser repartidas a lo largo de toda la red como si de un sistema distribuido se tratase. Precisamente, la mejor forma de enfrentarse a esta aspiración es el desarrollo de nuevas arquitecturas y protocolos que permitan la distribución de carga, o

resolución de problemas de forma local, en lo que podríamos definir como sistemas distribuidos de área extensa y banda ancha.

La *Figura 7.1* presenta un escenario p-p en la VPN en el que se pueden observar tres nodos AcTMs que soportan la arquitectura TAP. En el nodo intermedio se produce una congestión del buffer que es detectada por el algoritmo EPDR. El identificador de la PDU que experimenta la congestión es usado para solicitar la retransmisión, la cual es enviada al nodo emisor que atenderá la retransmisión de esa PDU siempre que tenga suficiente tiempo de inactividad. De este modo, la retransmisión no es realizada desde el extremo de la comunicación, sino desde el nodo intermedio con lo que conceptualmente ahorraremos la mitad del tiempo *RTT*.

Del mismo modo, en el nodo final se produce una nueva congestión, por lo que se genera una nueva célula RM con el valor del índice de la PDU congestionada que es atendida por el nodo intermedio. Si este nodo intermedio tiene tiempo para atender la retransmisión y además la PDU está aún en la memoria DMTE, se encargará de reenviar nuevamente la PDU hasta el nodo final. De este modo ahorramos nuevamente la mitad del tiempo *RTT* ya que la retransmisión no llega al nodo emisor, sino que es atendida localmente al conmutador en que se produce la congestión. Además, en este caso se consigue evitar el posible problema de implosión sobre el nodo emisor que delega su labor de retransmisión en un nodo intermedio y siempre más cercano al nodo congestionado.

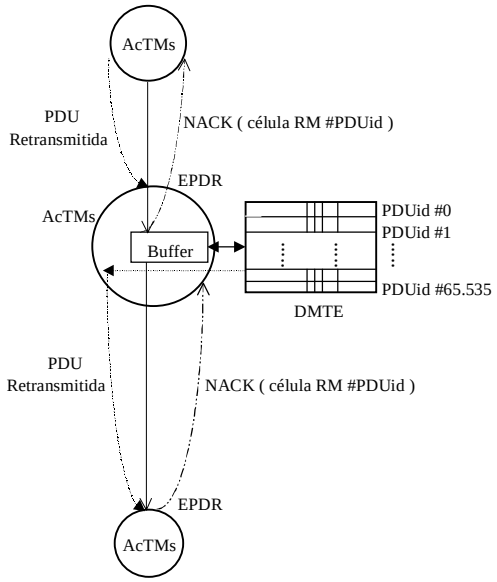

Figura 7.1. *Escenario p-p en VPN*

Podemos enfrentarnos a un nuevo escenario como el mostrado en la *Figura 7.2* en el que puede observarse una comunicación p-mp, donde además se representa la posibilidad de una VPN con nodos intermedios entre los destinatarios de la conexión y el nodo activo que se encarga de atender las retransmisiones antes de que lleguen a la fuente de tráfico. Como puede observarse en la *Figura 7.2*, los nodos no activos de la VPN que no soportan el protocolo activo TAP sólo van a atender los NACK como células RM nativas, por lo que la única labor que realizan estos conmutadores es la de reencaminar las peticiones de retransmisión en sentido al emisor. Si antes de llegar al emisor existe en la VPN un nodo AcTMs que aún contiene en su DMTE la PDU solicitada y tiene suficiente tiempo para atender la solicitud, se encargará de acceder a la memoria DMTE a través de los agentes software y reenviar la PDU nuevamente en el sentido al destinatario que la solicitó. En este caso también podemos ver cómo el ahorro de tiempo *RTT* puede ser importante y, sobre todo, al tratarse de una transmisión multipunto el problema de implosión será mucho mayor que en el caso del escenario de la *Figura 7.1*.

El último escenario al que nos podemos enfrentar es al representado en la *Figura 7.3*, en el que aparecen múltiples fuentes emisoras y múltiples destinatarios del tráfico. En realidad, el escenario mp-mp consiste en la yustaposición de *n* conexiones p-mp. En este caso el riesgo de implosión en los emisores es también evidente, por lo que la necesidad de disponer de un conmutador activo en el que los emisores puedan delegar las tareas de retransmisión es una opción interesante. La *Figura 7.3* introduce la posibilidad que entre el nodo activo intermedio y las fuentes de tráfico puedan existir en la red un número indeterminado de conmutadores no activos.

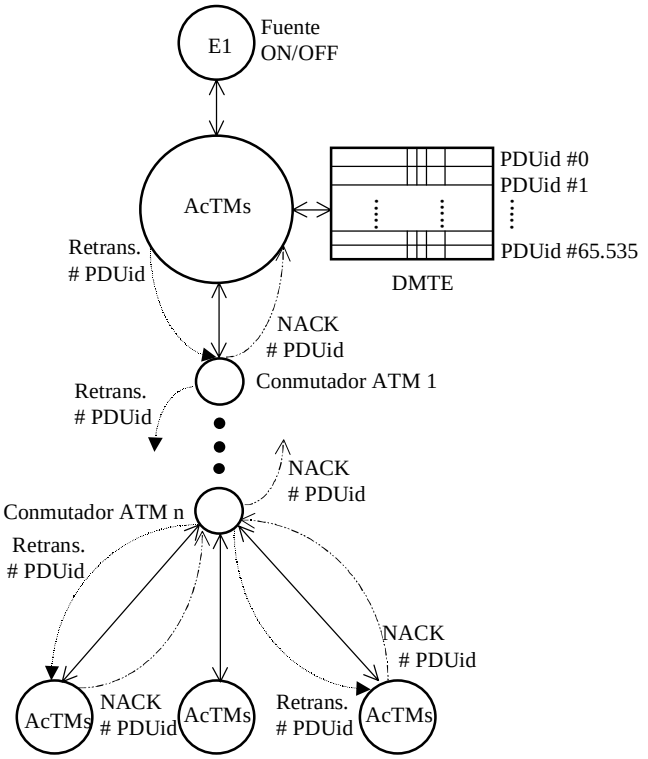

Figura 7.2. *VPN con un escenario p-mp y nodos no activos*

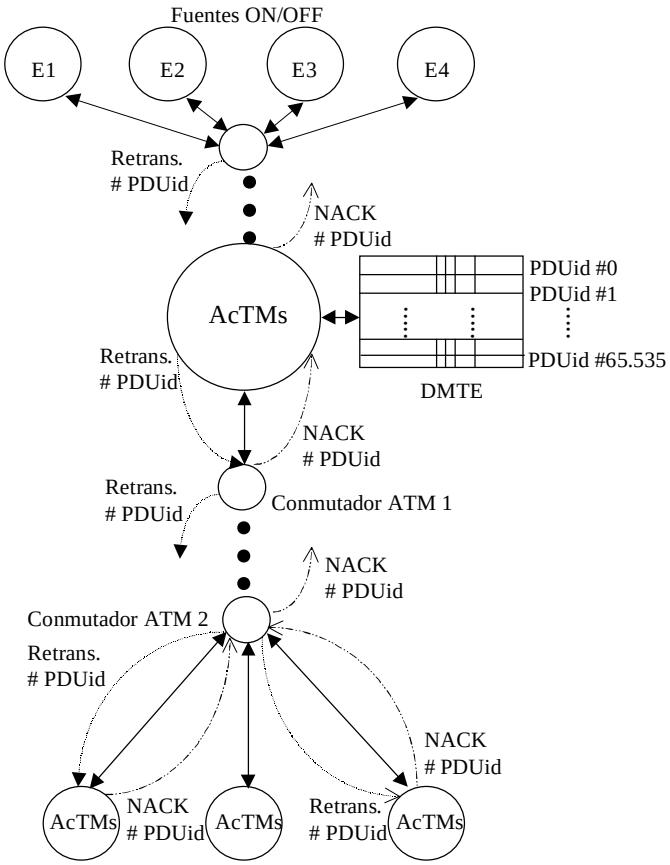

Figura 7.3. *VPN con escenario mp-mp*

A la vista de los tres escenarios comentados puede entenderse que el protocolo TAP está funcionando sobre toda la topología de la red, de forma que, por el hecho de disponer de un software distribuido (como el mismo protocolo y los agentes software) y un conjunto de escasos recursos hardware compartidos (como los propios conmutadores AcTMs y las memorias activas de cada uno de ellos), podemos decir que estamos ante un sistema distribuido que soporta el protocolo activo y distribuido TAP.

Podemos decir que el protocolo TAP cumple con todos los requerimientos de los algoritmos descentralizados propuestos en [2] que dan lugar a algoritmos distribuidos:

- Ningún componente de la VPN gestiona información completa sobre el estado completo del sistema. Es decir, cada conmutador activo almacena las PDU que le permite su propia DMTE, y no existe la necesidad de uso de ningún tipo de base de datos del sistema que pueda perjudicar el rendimiento final del mismo.
- Los nodos del sistema toman decisiones basándose únicamente en su información local. En este caso este aspecto es de crucial importancia para nosotros, ya que el sistema multiagente juega un importante papel en el sistema distribuido. Los agentes software locales a cada AcTMs aplican su propia inteligencia para actuar en su propio conmutador, aunque pueden aplicar la posibilidad de solicitar las retransmisiones a otros nodos hasta llegar el emisor.
- El fallo de uno de los componentes del sistema no pone en peligro la integridad del algoritmo TAP ya que la búsqueda de PDU congestionadas son locales a la congestión, pero pueden irradiarse en sentido a la fuente y a partir del nodo congestionado. Debemos destacar que ésta ha sido una de las precondiciones en nuestras investigaciones, ya que no podemos obligar a que todos los conmutadores de la red tengan que soportar el protocolo. En las figuras hemos comprobado cómo los nodos que no soportan TAP serán capaces de procesar las células RM nativas y estándares hasta llegar al emisor de la conexión. Si no hubiésemos previsto esta posibilidad nos encontraríamos con el inconveniente de poder asumir el coste económico de nuestra propuesta, porque obligaría a cambiar toda la equipación existente en una red para poder disponer de la GoS que ofrece TAP. Además, este hecho también impediría cumplir los requerimientos de escalabilidad, robustez, fiabilidad, disponibilidad, etc. de los sistemas distribuidos.
- No existe una asunción implícita de que exista un reloj global de todo el sistema. Aunque este requerimiento es más laxo en los sistemas distribuidos, en nuestro caso podemos garantizar que el protocolo no tiene ninguna necesidad de una sincronización de tiempos en el sistema. Los flujos dentro del sistema pueden seguir sus propios ritmos, sin poner en peligro el funcionamiento del algoritmo. En algunos casos puede ocurrir que las memorias DMTE no dispongan de las PDU solicitadas porque una fuente genere un tráfico demasiado fluido, pero es preferible asumir este riesgo antes que afectar al funcionamiento completo del sistema. Podemos destacar que en nuestro caso, el único mecanismo de control general en el sistema es la numeración secuencial de las PDU desde que salen del emisor. De este modo disponemos de una “ventana” de 65.535 PDU que pueden ser retransmitidas hasta que vuelve a comenzar un nuevo ciclo de PDU.

Sabemos ya que el principal objetivo de la arquitectura TAP es la de proporcionar soporte a un protocolo activo que aporta garantía de servicio a las conexiones de una red ATM. Para ello incorpora agentes software y un conjunto de refinados mecanismos de la ingeniería de protocolos con la intención de eliminar los problemas que ya han sido identificados como la implosión, la fragmentación de paquetes y el interleaving. De este modo, las propiedades principales de la arquitectura TAP, relacionadas con su carácter distribuido son:

- El carácter nativo ATM, que permite su completa integración con toda la tecnología ATM existente, ya que el protocolo es soportado sólo por los conmutadores AcTMs que pueden estar distribuidos por la topología de la VPN. Esto permite que no sea necesario cambiar todos los elementos de la red, sino sólo aquellos que se consideren oportunos y, sobre todo, aquellos conmutadores que se conozcan previamente como fuente potencial de congestiones.
- Eficiencia, debido a que TAP permite optimizar el rendimiento de la red consiguiendo que el goodput que suele degradarse en las situaciones de congestión sea mantenido en niveles mucho superiores a los que se obtendrían en el caso de no aplicar TAP. Además, debido a que se emplean los periodos de inactividad de las fuentes para recuperar las congestiones, se garantiza que el throughput general de la red no se vea comprometido ya que sólo se realizan retransmisiones cuando la red dispone de suficiente tiempo de silencio para no afectar a las fuentes de tráfico que puedan estar activas en cada instante de tiempo.

- Escalabilidad, ya que la complejidad y el rendimiento del sistema no se incrementan a medida que se incrementa el número de elementos que forman parte de nuestra arquitectura distribuida. Es decir, la incorporación de nuevos conmutadores AcTMs a la red no hace peligrar el rendimiento de la red, si no que se consigue aumentar el grado de GoS de las conexiones privilegiadas, o bien aumentar el número de conexiones privilegiadas que puede emplear el sistema.
- Justicia, pues los esquemas introducidos de colas justas permiten caracterizar el tráfico, consiguiendo aportar mayores anchos de banda a las conexiones que lo solicitan, y también aportando GoS a las conexiones que establecen su conexión como garantizadas o privilegiadas. No obstante, el protocolo implementado no descuida las situaciones de injusticia que puedan producirse, evitando que las conexione con menos necesidades puedan verse relegadas.
- Robustez, en el sentido que el SMA propuesto aporta robustez por sí mismo. Es decir, en situaciones en que algún agente del sistema deje de funcionar o, en el peor de los casos el SMA completo no actúe, no provocará ningún problema a la red. Únicamente dejarán de garantizarse todos los aspectos que TAP cuida, pero el funcionamiento general de la VPN no se ve amenazado por fallos de TAP.
- La simplicidad de la arquitectura es también otra de las ventajas de TAP. El SMA propuesto incorpora un total de cinco agentes software de los cuáles dos son programables y dinámicos en lo que respecta a su detección de eventos en tiempo real y durante la explotación de la red. Además, el esquema general de comunicación entre los agentes es también muy sencillo sin incorporar excesiva complejidad algorítmica, ni coste computacional por el acceso a complejas estructuras de datos, ni tampoco sobrecargar en las labores de coordinación entre los diferentes agentes del sistema.
- Estabilidad o equilibrio, ya que las simulaciones realizadas permiten demostrar que, ante la aparición de ruidos en el sistema (en nuestro caso, congestiones provocadas por tráfico de backgroud), éste es capaz de actuar poniendo en marcha el mecanismo de recuperación de PDU y, una vez, abandonada la situación de congestión el sistema es capaz de volver o converger a su punto de equilibrio o estabilidad como si de un sistema dinámico se tratase.
- Economía, ya que la equipación hardware que hay que incorporar a los conmutadores para que soporten el protocolo propuesto no va más allá de una simple memoria de almacenamiento de PDU. Además, se propone también la implementación de la mayor parte de los esquemas software desarrollados mediante hardware para conseguir un mayor grado de integración y funcionamiento.

Para conseguir todas estas características se requiere de un sistema claramente distribuido, en el que no sea necesaria ninguna visión ni control central de la red ni de sus datos. No obstante, como lo que se persigue es un óptimo funcionamiento de la red que incorpora TAP es importante destacar que ésta permite al operador de la red su intervención para ajustar los parámetros de las conexiones en las situaciones en las que se puede perder el estado de equilibrio del sistema. Es decir, aunque TAP es capaz de resolver el problema de las pérdidas de PDU, el SMA permite al operador de la red la intervención (introduciendo órdenes y objetivos) para intentar evitar las situaciones de congestión generando acciones de los agentes programables del sistema.

7.5. SISTEMAS DISTRIBUIDOS FIABLES CON CORBA

CORBA (Common Object Request Broker Architecture), se ha convertido en el estándar para diseñar sistemas abiertos distribuidos [11], también en entornos de comunicaciones. En sus inicios CORBA estuvo principalmente pensado para la distribución de información en conexiones p-p y no ofrecía soporte para el desarrollo de aplicaciones fiables con comportamiento predecible en sistemas distribuidos. Sin embargo, no han tardado en aparecer extensiones como [9,12] pensadas para que aplicaciones más sofisticadas como teleconferencia, VoD y otros servicios para grupos de usuarios soporten fiabilidad y tolerancia a fallos.

La Programación Distribuida Orientada a Objetos (PDOO) puede verse como una variante del modelo cliente-servidor, y es destacable que el diseño orientado a objetos y la implementación de sistemas distribuidos aportan importantes ventajas como la separación del servicio garantizado de los objetos de la propia tecnología de implementación. Sin embargo, el soporte de aplicaciones distribuidas fiables supone importantes esfuerzos ante lo impredecible de los retardos en las comunicaciones y de los fallos parciales en las redes como pueden ser los problemas debidos a las congestiones que estudiamos en esta tesis.

La fiabilidad en CORBA ha sido conseguida con el desarrollo de abstracciones para la gestión de grupos multicast que han complicado de forma importante los sistemas distribuidos. La fiabilidad de los grupos multicast requiere que las solicitudes sean confirmadas por todos los miembros de los grupos. Por esto los

posible fallos pueden conducir al bloqueo de algunos clientes que esperaran indefinidamente la respuesta de miembros de los grupos que están experimentando problemas. Para solventar estas situaciones se han propuesto soluciones como las solicitudes atómicas[5], que permiten mantener la consistencia de los grupos multicast cuando se produce la detección y propagación de fallos.

7.6. CONCLUSIONES

En este capítulo hemos querido destacar las principales características de los SD con la intención de incorporar las ventajas de la computación distribuida a nuestras investigaciones. Hemos podido ver cómo la propia definición del concepto de SD implica aspectos hardware y software que, en nuestro caso concreto, se resuelven con la propuesta de la arquitectura TAP sobre conmutadores activos AcTMs. Los conmutadores activos constituyen una VPN distribuida (por el carácter distribuido del protocolo TAP) y, por otro lado, contamos con la equipación en esta arquitectura de las técnicas activas que aportan los agentes software del SMA-TAP. Además, todo el sistema distribuido resultante es controlado mediante el protocolo TAP visto como un algoritmo descentralizado. Toda esta propuesta nos permite argumentar que TAP constituye un SD que es capaz de garantizar características propias de los SD como la escalabilidad, el rendimiento, la fiabilidad, la flexibilidad, la robustez y la transparencia para los usuarios que requieran las conexiones garantizadas en la VPN propuesta. Por otro lado, CORBA aparece como la forma de incorporar la distribución de servicios de una forma estándar que soporta además la tecnología de los agentes software. Aunque nuestras implementaciones no usan CORBA, hemos identificado esta posibilidad como una futura línea de trabajo que permita la mejora de TAP.

REFERENCIAS

[1] George Coulouris, Jean Dollimore, and Tim Kindberg, "Distributed systems: concepts and design," *Ed. Addison-Wesley* (1994).

[2] Andrew S. Tanenbaum, "Distributed Operating Systems," *Ed. Prentice Hall International Editions*, (1995).

[3] William Stallings, "SNMP, SNMPv2, SNMPv3 and RMON 1 and 2," *Ed. Addison-Wesley*, (1999).

[4] Pere Vilà and Josep L. Marzo, "Scalability Study and Distributed Simulations of an ATM Network Management System based on Intelligent Agents," *SPECTS'2K 2000 SCS Symposium on Performance Evaluation*, (2000).

[5] L.C. Lee, H. S. Nwana, D. T. Ndumu and P. De Wilde, "The stability, scalability and performance of multi-agent systems," *http://www.labs.bt.com/projects/agents/publish/papers.htm*

[6] Jiann-Liang Chen, Shingfeng Lin, Ming-Yi Lee, "Distributed fault management over ATM networks," *Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp.196-201. (Oct. 1997).

[7] Simon Steward and Steve Appleby, "Mobile software Agents for Control of Distributed Systems Based on Principles of Social Insect Behaviour," *Proceedings Conference Singapore ICCS'94*, pp. 549-553, (1994).

[8] Gwan Joong Kim, Young Sun Kim and Hyeong Ho Lee, " A design of Management System for ATM Switches Using Mobile Agent Concept," *Global Telecommunications Conference, GLOBECOM 1998. The Bridge to Global Integration. IEEE*, pp. 1694-1698 vol.3, (1998).

[9] Robert Orfali, and Dan Harkey, "Client/Server Programming with Java and CORBA," *Ed. John Wiley and Sons*, (1998).

[10] Paul Ferguson and Geoff Huston, "White paper: What is a VPN " *http://www.employees.org:80/~ferguson/* (2000).

[11] Object Management Group, "The Common Object Request Broker: Architecture and Specification," (1995).

[12] Sean Landis, and Silvano Maffeis, "Building Reliable Distributed systems with CORBA," *http://citeseer.nj.nec.com/landis97building.html*, (1997).

[5] La atomicidad en las solicitudes garantiza que las solicitudes enviadas a los grupos de objetos serán recibidas por todos los miembros del grupo o por ninguno.

CAPÍTULO 8

MOTIVACIONES GENERALES

8.1. INTRODUCCIÓN

El principal objetivo de este capítulo es el de presentar las motivaciones generales de nuestra investigación, para poder argumentar el verdadero valor de las aportaciones realizadas. En la actualidad las redes ATM se están usando como la tecnología para soportar todo tipo de tráfico, pero con un destacable predominio de los protocolos TCP/IP. Por esto vamos a presentar los beneficios que nuestro mecanismo de recuperación de congestiones puede aportar, no sólo al trafico ATM nativo, sino también al tráfico generado por las fuentes TCP/IP.

El protocolo TCP se ha convertido en los últimos años en el estándar de comunicaciones de datos. Este es un protocolo fiable de la capa de transporte de la arquitectura TCP/IP, que usa control de error y control de flujo basados en ventana y se encarga del enrutamiento de paquetes en internet con control extremo-a-extremo [1].

Como ya hemos comentado en el *Capítulo 2*, existen numerosas investigaciones para conseguir integrar dos tecnologías tan diferentes como ATM y TCP/IP. Sin embargo, la integración de ambas se ha demostrado [2] con un pobre resultado en cuanto al comportamiento del throughput de TCP sobre ATM. Mientras ATM es, en líneas generales, una tecnología orientada a conexión, de conmutación de células de 53 octetos y de tamaño uniforme, TCP e IP se basan en mecanismos de enrutamiento de segmentos y datagramas de tamaño variable, generalmente de 1.500 octetos y mucho mayores de 48 octetos que es la unidad de datos básica de ATM. Así, cuando una fuente TCP emplea una red ATM como medio de transporte, los segmentos deben ser adaptados a las unidades de procesamiento que ATM es capaz de conmutar. Los segmentos TCP son encapsulados en PDU de la capa AAL-5, que serán luego segmentadas por la capa ATM en células de 48 octetos que son conmutadas en dirección al receptor del tráfico. Podemos decir, por tanto, que el pobre rendimiento de TCP se produce por su propio dinamismo que es a menudo empeorado cuando se implementa sobre ATM, y por la fragmentación de paquetes que ocurre cuando un paquete TCP fluye dentro de una conexión VPI/VCI a través AAL-5. La segmentación de AAL-5 es necesaria porque el tamaño típico de los segmentos TCP es mucho mayor que el de las células ATM. Además, el tráfico TCP es generalmente a ráfagas por lo que no permite su caracterización para conseguir predecir su comportamiento a priori. TCP transmite los segmentos tan rápido como es posible y usando una ventana para dar solución a las congestiones.

Las características que acabamos de comentar provocan un efecto bastante devastador en el throughput cuando los segmentos TCP atraviesan conmutadores ATM con tamaño de buffer mucho menor que el tamaño de ventana de TCP. Por esta causa las células se pierden y acaban generándose retransmisiones por timeout. Además, la pérdida de una sola célula dará como resultado la pérdida de un segmento TCP en el receptor de la comunicación, por lo que solicitará una retransmisión al emisor que debe encargarse de reenviar nuevamente el segmento perdido completo, en lugar de la célula errónea. Este mecanismo acaba degenerando el rendimiento de la red con un considerable desaprovechamiento del goodput de la misma. Como ATM no dispone de control de acceso al medio (MAC) el throughput cae considerablemente si la red comienza a experimentar congestiones.

TCP es un protocolo de transporte orientado a conexión que basa su fiabilidad en retransmisiones extremo-extremo cuando aparecen problemas. En este caso hay coincidencias con ATM, y por esto consideramos que nuestra propuesta de GoS acaba aportando ventajas no sólo al tráfico ATM nativo, sino también al predominante TCP. Para demostrar nuestra tesis vamos a presentar una serie de simulaciones que demuestran el comportamiento de TCP cuando se enfrenta a las congestiones. Para ello vamos a usar el simulador NS (Network Simulator) con el que estudiamos el mecanismo de ventana de TCP fijándonos en los efectos del umbral, de la ventana de congestión, de la probabilidad de pérdidas de segmentos, de los retardos, y en las consecuencias que todo esto acaba teniendo sobre el throughput. Veremos cómo el goodput acaba degenerándose cuando aparecen congestiones en los escenarios simulados. Si introducimos TAP en la red conseguiremos mejorar sustancialmente el goodput de las transmisiones TCP ya que logramos eliminar las retransmisiones e-e y además las labores de autoajuste de las ventanas de congestión en las fuentes TCP.

En primer lugar comentamos las características generales de TCP, para pasar después a las simulaciones con NS (Network Simulator) [3] en varios escenarios. La siguiente sección propone la inclusión de TAP en la red en un escenario IPoATM. Terminamos el capítulo con un apartado de conclusiones de las motivaciones generales de nuestro trabajo que se propone mantener el throughput en TCP sobre ATM considerando los aspectos de justicia, de ahorro del RTT, de la fragmentación de paquetes, y de la implosión, todo ello apoyándonos en el SMA-TAP.

8.2. FUNCIONAMIENTO DE TCP

El protocolo TCP es en la actualidad un conjunto de algoritmos que envían paquetes a la red sin ningún tipo de reserva previa, pero que son capaces de reaccionar ante determinados eventos en la misma. Entre esos algoritmos destaca el *Control de Congestión* y el de *Recuperación de Segmentos Perdidos*. Para poder llevar esto a cabo, cada emisor mantiene dos ventanas: RWND (*Receiver Window*) y CWND[1] (*Congestion Window*). Cuando se envían paquetes se usa la ventana más pequeña de estas dos.

El mecanismo de control de congestión en TCP tiene dos fases diferentes: *Slow Start* y *Congestion Avoidance*. Al iniciar una conexión, o al reiniciarse por el envío de un segmento perdido, el tamaño de la ventana de congestión es puesta a 1 paquete, y después es aumentada al doble en cada ACK recibido desde el receptor en el tiempo RTT. El mecanismo *Congestion Avoidance* se encarga de la violación que puede producirse cuando el tiempo de retransmisión es demasiado corto. Durante esta fase, CWND es incrementada linealmente al contrario de su crecimiento exponencial durante la fase *Slow Start*.

Se emplea un tercer parámetro que es el THRESHOLD, usado para arrancar la fase de congestión y que es inicializado a 64 Kb. Cuando aparece congestión, (por ejemplo, detectada por un timeout) el valor de THRESHOLD es puesto a la mitad del valor actual de la ventana CWND, la ventana CWND es puesta a 1 y la fase de *Slow Start* es reiniciada de nuevo.

La implementación original de TCP debida a Jacobson [4] ha dado lugar a diversas variantes que son conocidas como distribuciones Tahoe y Reno. Posteriormente han aparecido versiones más actualizadas y cuidadas de la implementación de TCP como TCP Vegas [5] o Selective Acknowledgement. En general, los algoritmos de control de congestión de TCP se han preocupado [6] por diversos aspectos concretos como: rendimiento, escalabilidad, justicia, complejidad y compatibilidad con la tecnología actual.

Por tanto, las fuentes TCP determinan la velocidad de envío de datos usando una ventana de control de flujo. El emisor envía una ventana de paquetes por cada RTT. TCP ajusta su tamaño de ventana para reflejar las condiciones de la red según lo siguiente: 1) cada vez que TCP envía una ventana de paquetes, éste incrementa el tamaño de la ventana en un paquete; 2) cada vez que la red tira un paquete, TCP reduce el tamaño de la ventana a la mitad.

TCP puede detectar la pérdida de paquetes rápidamente usando "*fast retransmit*" tan pronto como la ventana sea mayor de tres paquetes. Si falla el "*fast retransmit*" TCP cae en una retransmisión conservadora basada en timeout de un segundo o más. Así, la pérdida afecta al retardo de dos formas: 1) decrementando el tamaño de la ventana de TCP y la velocidad del envío; y 2) forzando a caer a TCP en *timeouts*. La referencia [7] muestra resultados del efecto de la perdida de paquetes en los retardos de los paquetes provocadas por estas cuestiones, de forma que puede verse claramente cómo, a medida que se incrementa la pérdida de paquetes, la eficiencia de TCP cae sustancialmente. Puede comprobarse cómo, también en el caso de elevadas velocidades de transmisión, se acaban generando comportamientos erráticos e injustos en el retardo que no se ajustan a los valores medios deseables.

[1] El control de flujo en TCP se basa en el tamaño de CWND en cada envío de segmentos. Las cabeceras de cada segmento disponen de un campo de 16 bits que indica el tamaño de ventana y lo limitan a un máximo de 65.535 bytes.

Los siguientes son los aspectos básicos de funcionamiento de TCP:

- Segmento: designa de forma genérica los paquetes TCP/IP, tanto los de datos como los ACK. Generalmente las unidades de transferencia de IP se designan con el nombre de datagramas, mientras las del protocolo TCP se conocen como segmentos.
- CWND: representa la ventana de congestión, y no es otra cosa que una variable que limita la cantidad de datos que TCP puede enviar. Su tamaño varía dependiendo de las condiciones de la red, de forma que si la red no descarta paquetes por congestión, el tamaño de esta ventana aumenta permitiendo aumentar también la velocidad de transmisión de las fuentes de tráfico.
- INITIAL_WINDOW: es el valor con que se inicia la ventana de congestión CWND.
- SMSS: expresa la cantidad máxima de datos que una fuente de tráfico TCP puede enviar.
- RWND: es la cantidad máxima de datos que puede recibir un receptor de tráfico TCP.
- RTT: es el *round trip time*, es decir el tiempo que transcurre desde que un segmento sale del emisor hasta que éste recibe la confirmación de que ha sido recibido por el receptor. En realidad, el RTT determina la velocidad de transmisión de TCP, ya que el emisor envía cada RTT el tamaño de datos determinado por la ventana CWND.
- CURRENT_WINDOW: representa la cantidad de información que envía el emisor cada RTT. Esta ventana toma como valor el más pequeño de CWND o RMSS.
- SSTHRESH: es una variable que se usa para determinar qué algoritmo de control de congestión se debe usar. Como ya se ha indicado, estos dos algoritmos son *Slow Start* y *Congestion Avoidance*. Cuando CWND < SSTHRESH se emplea el algoritmo *Slow Start*, mientras que cuando CWND ≥ SSTHRESH se aplica el control de congestión determinado por *Congestion Avoidance*.

Por tanto, una fuente TCP establece la cantidad de datos que envía usando una ventana (Window Flow Control) y envía una ventana de segmentos por cada RTT. TCP ajusta el tamaño de dicha ventana dependiendo de las condiciones de la red. Así, el tamaño de CWND se incrementa en el doble de segmentos por cada ACK recibido si estamos en *Slow Start*, y se incrementa en 1/CWND por cada ACK recibido en *Congestion Avoidance*. Todo esto ocurre en una conexión en la que no hay descartes de paquetes, y lo ilustramos con la *Figura 8.1*, donde podemos observar la ventana de congestión sin pérdidas que acabamos de comentar. La ventana de congestión aumenta exponencialmente mientras su tamaño es menor que SSTHRESH, esto es debido a que se está usando el algoritmo *Slow Start* aumentando progresivamente el número de segmentos (1, 2, 4, 8, 16...) a medida que se van recibiendo los ACK. Pero cuando el tamaño de CWND se iguala al valor de SSTHRESH entra en acción el control de congestión de *Congestion Avoidance*. A partir de este momento la ventana incrementa en 1/CWND por cada ACK, dando lugar a un crecimiento lineal de la ventana CWND. Podría decirse que el algoritmo *Slow Start* es usado por TCP para probar la capacidad de la red (de la que se desconoce su capacidad) y la cantidad de segmentos que puede soportar sin congestionarse. En cuanto se acerca una posible congestión, TCP cede control al algoritmo *Congestion Avoidance* que permite el incremento lineal de CWND hasta que la congestión es detectada.

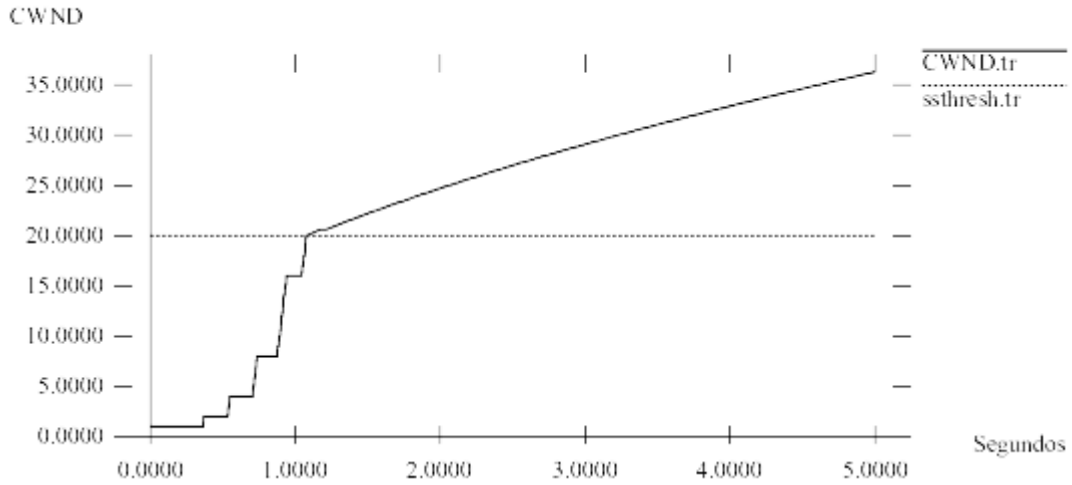

Figura 8.1. *Evolución de la ventana de congestión sin pérdidas*

Comentario aparte merecen las novedosas investigaciones aportadas por [8] donde se demuestra la naturaleza caótica del control de congestión de TCP en función de aplicaciones concretas de las redes TCP/IP. Del mismo modo es también destacable la aportación al buen funcionamiento de TCP del concepto de "*pacing*" o espaciado de los datos aportados por [9] a las fuentes TCP cuyos mecanismos de control de congestión acaban degenerando en tráfico a ráfagas que provocan la pérdida general de eficiencia de la red.

8.2.1. REACCIÓN ANTE CONGESTIONES

Cuando la capacidad de transmisión de la red es inferior a la cantidad de información que desea transmitirse, la propia red comenzará a descartar segmentos en un intento porque los emisores disminuyan la cantidad de información que generan. TCP detecta que un segmento ha sido descartado por congestión si el número de ACK repetidos recibidos es 3, o cuando el timeout de las retransmisiones ha expirado. TCP reaccionará reinicializando el valor de SSTHRESH a la mitad de la ventana, reduce CWND al valor determinado por INITIAL_WINDOW, reduciendo así la cantidad de segmentos que envía a través de la red. En suma, se opera según el siguiente algoritmo:

- Inicialización de CWND a un segmento, y del umbral SSTHRESH a 65.535 bytes.
- La rutina de salida de TCP nunca envía más datos del menor de los valores CWND y RMSS.
- Cuando se produce la congestión (timeout o ACK duplicados) SSTHRESH es reducido y toma el valor mínimo $min(\frac{CurrentWindow}{2}, 2)$ y CWND se inicializa a INITIAL_WINDOW.
- Al recibir un nuevo ACK en el emisor, se incrementa CWND dependiendo del algoritmo de control de congestión que se esté usando en ese instante y según lo ya comentado.

La *Figura 8.2* representa la situación comentada y podemos observar la evolución de la ventana de congestión con pérdidas en la red debidas a congestiones.

Tanto la *Figura 8.1* como la *Figura 8.2* se han obtenido con la simulación de la misma topología sobre NS. La topología es presentada en la *Figura 8.3*, donde pueden observarse los 7 nodos que intervienen. Cada uno de los enlaces tiene un ancho de banda de 1 Mbps, un delay de 10 ms. y usan DropTail como tipo de cola. El escenario de simulación está compuesto por los tres agentes siguientes:

- tcp (Agent/TCP) actúa como el emisor de la conexión y está asociado al nodo 0. Es el agente a monitorizar y del que se obtienen los valores de CWND y SSTHRESH.
- tcpSink (Agent/TCP) es el agente que actúa como receptor de la conexión, que envía los ACK al agente tcp. Este agente está asociado al nodo 6.
- ap (Application/Traffic/CBR), es el agente que realmente genera el tráfico, y está asociado al agente tcp (nodo 0).

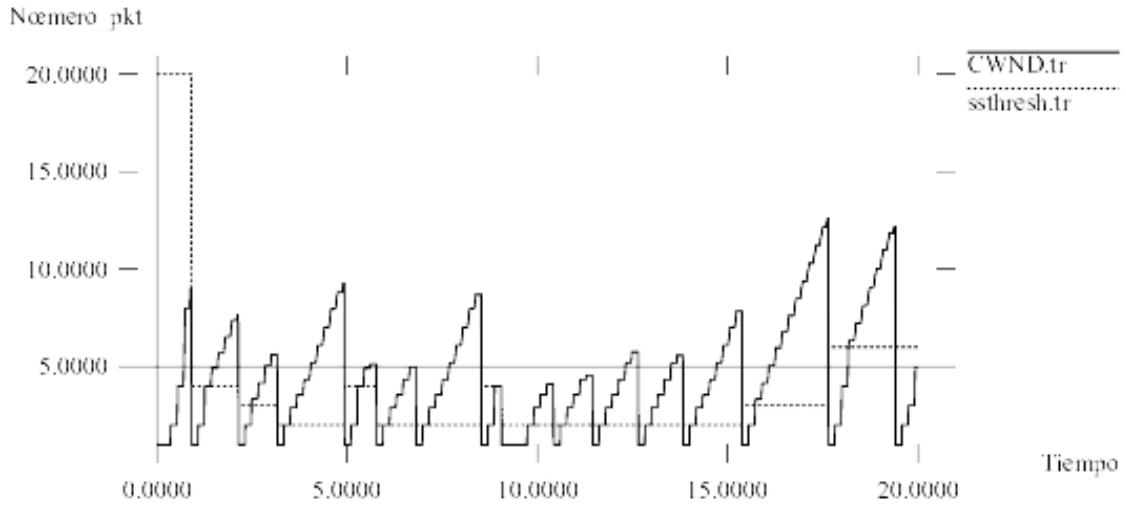

Figura 8.2. *Evolución de CWND y de SSTHRESH en una red congestionada*

La diferencia de los dos escenarios simulados está en el hecho que mientras la *Figura 8.1* muestra el comportamiento de TCP sin pérdidas de segmentos, la *Figura 8.2* introduce la pérdida de paquetes TCP. Para ello, en el segundo caso se ha introducido la probabilidad de pérdida mediante modelos de error que nos han permitido definir una probabilidad de pérdida de 0,02 en los enlaces 2-3 y 3-4. En la *Figura 8.3* puede observarse cómo el nodo 3 ha perdido un paquete que ha sido descartado por la red.

En el caso de la *Figura 8.2* se ha empleado un tiempo de simulación superior (20 s.) para poder comprobar con claridad el efecto de las pérdidas sobre la ventana de congestión que es reducida a 1 en múltiples ocasiones para solventar el problema de las congestiones. Destacamos que nuestra propuesta TAP intenta solventar estos problemas de pérdidas que afectan, tanto a la reducción de la ventana, como a la posterior retransmisión de las pérdidas extremo-a-extremo. Así, la fuente no se verá obligada a reducir su velocidad de envío tan a menudo como muestra la *Figura 8.2* y, sobre todo, cuando aparezcan las congestiones, éstas serán resueltas localmente entre los nodos afectados, siempre que sean nodos AcTMs.

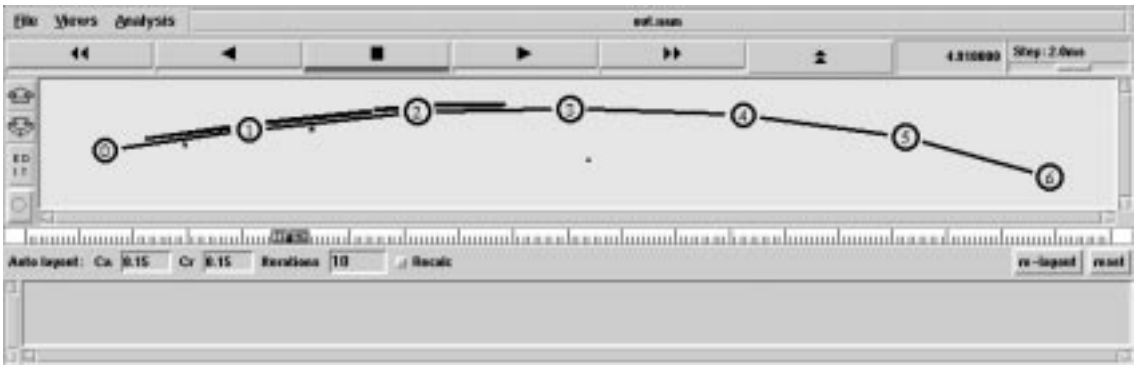

***Figura 8.3**. Topología de simulación NS para las Figuras 8.1 y 8.2*

8.2.2. THROUGHPUT Y PROBABILIDAD DE PÉRDIDAS

Puede estimarse el comportamiento de TCP considerando que el enlace sobre el que se usa cumple las siguientes características:

- El valor de *RTT* es constante.
- El ancho de banda del enlace es más que suficiente para soportar la conexión TCP.
- Existe una probabilidad de pérdidas en la red que es aleatoria y constante y la representamos por *P*. Suponemos por tanto que el emisor es capaz de enviar *1/P* paquetes consecutivos antes de que un segmento TCP sea descartado provocando la congestión y el recorte de CWND justo a la mitad de su tamaño actual.

Podemos destacar que la ventana de congestión CWND dibuja el gráfico periódico presentado en la *Figura 8.4* [10].

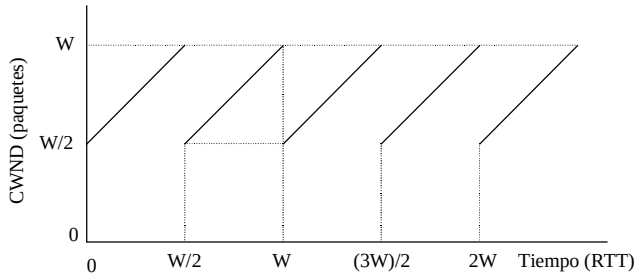

Figura 8.4. Evolución de la ventana CWND de TCP con pérdidas periódicas

A la vista de la *Figura 8.4*, y suponiendo que el tamaño máximo de la ventana CWND es de *W* segmentos, entonces según la definición que hemos hecho del algoritmo *Congestion Avoidance*, el valor mínimo de CWND es de *W/2* segmentos. Como sabemos que el receptor devuelve un ACK por cada segmento recibido, entonces CWND se incrementa en un segmento por cada RTT, o bien cada *(RTT*(W/2))*

segundos. A partir de esto se deduce que el total de datos entregados a la red en cada ciclo es precisamente el área de la superficie situada por debajo del diente de sierra perfecto que se observa en la *Figura 8.4*. Así, este volumen de datos o número de segmentos TCP entregados en cada ciclo puede ser calculado por la expresión

$$(\tfrac{W}{2})^2 + \tfrac{1}{2}(\tfrac{W}{2})^2 = \tfrac{3}{8}W^2 \tag{1}$$

Supóngase ahora una red sin pérdidas con un RTT constante porque tiene suficiente ancho de banda y con una baja carga total que nunca llene las colas. Pues bien, según [10] puede aproximarse la pérdida de paquetes aleatoria mediante una probabilidad constante *P* que asuma que el enlace entrega aproximadamente *1/P* paquetes consecutivos seguidos de un descarte, todo esto sin considerar los datos que se transmiten durante las recuperaciones de paquetes. Tenemos por tanto dos aproximaciones a la entrega de paquetes, la expresión (1) y *1/P*, por lo que uniendo ambas y despejando *W* obtenemos,

$$W = \sqrt{\tfrac{8}{3p}} \tag{2}$$

Por otro lado, podemos aplicar estos datos conocidos sobre la siguiente expresión que calcula el ancho de banda (donde *MSS* es el máximo tamaño de segmento TCP) transmitido,

$$AB = \frac{datos/ciclo}{tiempo/ciclo} = \frac{MSS * \frac{3}{8}W^2}{RTT * \frac{W}{2}} = \frac{MSS/P}{RTT * \sqrt{\frac{2}{3P}}} \tag{3}$$

Se puede reestructurar la expresión (3) agrupando el término constante $K = \sqrt{\frac{3}{2}}$; llegando a,

$$AB = \frac{MSS}{RTT}\frac{K}{\sqrt{P}} \tag{4}$$

La fórmula (4) expresa el ancho de banda de la red y nos sirve así para aproximar de forma sencilla el funcionamiento de TCP tras haber realizado algunas simplicaciones como las explicadas. El artículo [10] presenta otras referencias con diversas aproximaciones para el valor de la constante *K* que, indiferentemente de su valor, puede servir para considerar que su valor es siempre menor que 1, con lo cual podemos quedarnos finalmente con la siguiente fórmula (5) como una buena expresión del throughput de TCP,

$$AB < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{P}} \tag{5}$$

Por nuestra parte hemos analizado también el comportamiento de TCP en diversas situaciones y queremos plantear el comportamiento y efecto del throughput estudiado con respecto a la probabilidad de pérdida de paquetes. Para ello reorganizamos la formulación (5) del algoritmo *Congestion Avoidance* de TCP que expresa el comportamiento "*steady state*" de TCP bajo condiciones ligeras de carga y para una pérdida de paquetes moderada en su expresión general,

$$TH = \frac{MSS}{RTT\sqrt{P}} \tag{6}$$

Si en la ecuación (6) *TH* representa el throughput y consideramos como constantes los valores de *MSS* y *RTT*, podemos obtener la fórmula (7) donde comprobamos cómo el throughput es inversamente proporcional a la probabilidad de pérdida de paquetes,

$$TH = \frac{K}{\sqrt{P}} \tag{7}$$

La expresión (7) nos permite estudiar, por tanto, el comportamiento que va a experimentar el throughput a medida que se produzcan pérdidas de paquetes en routers congestionados. Dando valores a la *P* podemos obtener la representación gráfica de la *Figura 8.5* donde se observa una pendiente logarítmica negativa que indida la caida de la efectividad de la red a medida que se pierden segmentos TCP.

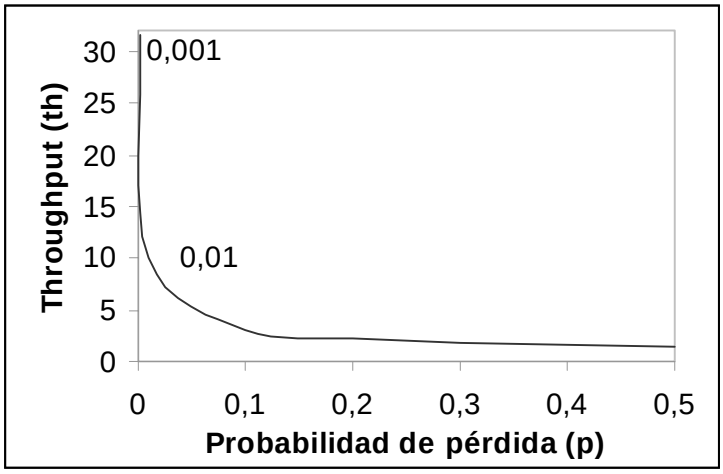

Figura 8.5. *Efecto de la probabilidad de pérdida sobre el throughput*

Como podemos observar en la *Figura 8.5*, entre una probabilidad de pérdida de 10^{-3} y 10^{-1} , la degradación del comportamiento de la red cae logarítmicamente, mientras a partir de valores estables de pérdidas de paquetes el comportamiento de la red es prácticamente constante. El problema lo encontramos en el hecho que TCP dobla los intervalos de los tiempos de retransmisión entre pérdidas sucesivas de paquetes. Es decir, si en la ecuación (6) no consideramos constante el valor de *RTT* nos encontraremos con que su efecto sobre la *Figura 8.5* será aun más negativo para la eficiencia de la red.

De la formula (7) puede obtenerse una nueva función que aproxima el tamaño de la ventana *W* que emplea TCP cuando se tiene una velocidad media de pérdidas *P*. Así, ajustando el valor de la constante *K* de la fórmula (4), obtenemos la siguiente expresión que es resultado de la adaptación de (7) en [10] y de las propuestas realizadas en [11],

$$W = \frac{0{,}866}{\sqrt{P}} \tag{8}$$

En nuestro caso particular, para el valor de la constante *K* calculamos $\sqrt{3/4}$ suponiendo que se aplica la estrategia retardada en los ACKs en lugar de aplicarla sobre cada uno de los paquetes, y según el planteamiento de pérdidas periódicas, en lugar de partir de un plantemiento de pérdidas aleatorias.

Así, y en línea con [7], podemos comprobar que la ecuación (8) puede verse desde dos puntos de vista diferentes. En primer lugar, la red descarta los paquetes a una velocidad independiente de la actividad del emisor, por lo que la fórmula expresa entonces la forma en que el emisor es capaz de reaccionar. En segundo lugar, si consideramos que la red es capaz de almacenar sólo un número determinado de paquetes, la ecuación puede entenderse como la velocidad que la red debe imponer para que la ventana de TCP quepa en esa capacidad de almacenamiento de la red. Según todo esto, podemos reorganizar la fórmula (8) y obtendremos la velocidad media de pérdidas *P* según la siguiente expresión,

$$P = \frac{0{,}75}{W^2} \tag{9}$$

La ecuación (8) puede entenderse como si la red descartase un porcentaje de segmentos independientemente de las acciones que realice la fuente. Es decir, describe la forma en que va a reaccionar el emisor. Para estudiar este comportamiento hemos simulado con NS el escenario de la *Figura 8.6*. Se han empleado enlaces de 2 Mbps de ancho de banda, con retardos de 10 ms. y la cola es *DropTail*. Además, se ha asociado a cada enlace una probabilidad de pérdida inicial de 0,001 que es incrementada el 5% cada 5 segundos en todos los enlaces.

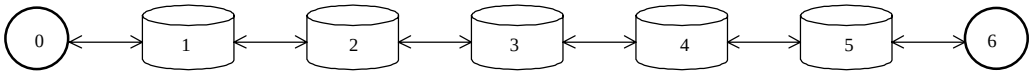

Figura 8.6. *Escenario para estudiar el funcionamiento de TCP*

El objetivo de este escenario es estudiar el comportamiento del ancho de banda con respecto a la probabilidad *P* de errores. Como resultado hemos obtenido el gráfico de la *Figura 8.7* en el que nuevamente podemos comprobar cómo se cumple el comportamiento macroscópico que indicaba la intuitiva *Figura 8.1*. Podemos ver cómo en la *Figura 8.7* la probabilidad de pérdida *P* acaba determinando el ancho de banda de la fuente TCP, como también lo indica intuitivamente la fórmula (8) que obtuvimos anteriormente. A medida que aumentamos la probabilidad de pérdida disminuye el ancho de banda logarítmicamente hasta acercarse a una evolución lineal cuando la probabilidad de pérdidas desciende, que es lo que intenta TAP.

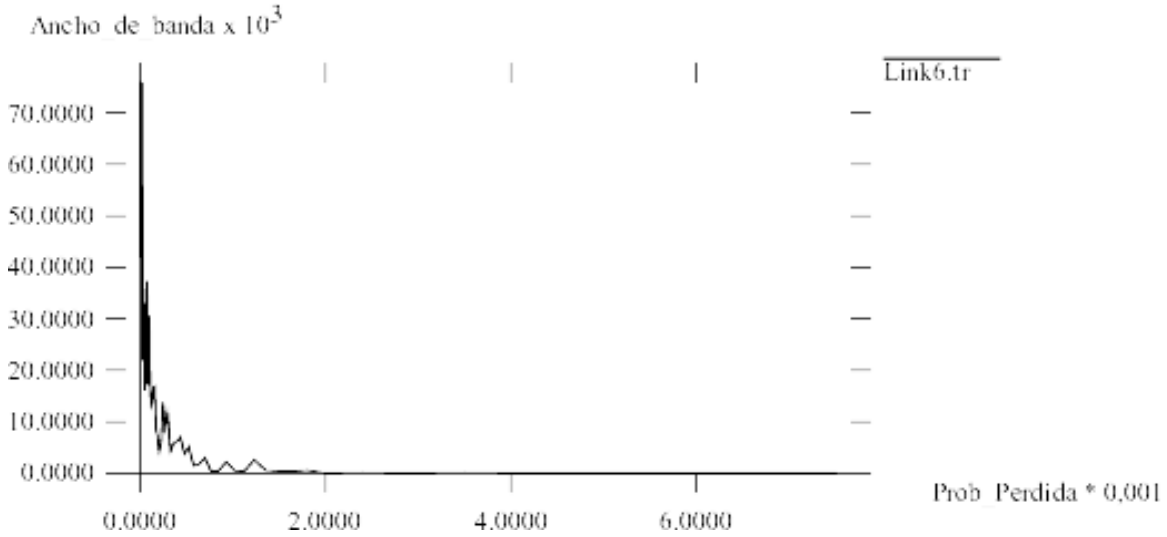

Figura 8.7. *Simulación de la ecuación (4) con NS*

8.2.3. ADAPTACIÓN DEL ANCHO DE BANDA DEL EMISOR

Una consideración importante a tener en cuenta en el caso de TCP es que el ancho de banda disponible para el emisor es adaptado por la red de acuerdo a su propia capacidad. Si consideramos que la red dispone de una capacidad limitada para la transferencia de un número concreto de paquetes, la ecuación (8) puede expresar la probabilidad de pérdida que la red debe imponer para que el emisor del tráfico TCP se adapte a la capacidad de transferencia de la red en cada momento. Así la ecuación (8) la hemos reestructurado para entender este planteamiento, dando lugar a la fórmula (9) vista anteriormente.

Hemos analizado una aplicación de la formula (9) consistente en un emisor TCP enviando tráfico a través de un router que dispone de una cola de longitud máxima de 8 paquetes. El router acaba generando una situación de congestión, por lo que la ventana CWND del emisor debe variar entre 8 y 4 paquetes con una media de 6 paquetes. Para lograr esta situación con la ecuación (9), según [7,10], el router debe descartar el 2,1% de los paquetes del emisor TCP. Este es el modo en el que la red usa la probabilidad de pérdida *P* para indicar a TCP el tamaño correcto de la ventana de congestión. Este estudio lo realizamos mediante la simulación del escenario presentado en la *Figura 8.8*.

Este escenario consta de una topología con 4 nodos, donde los enlaces 0-1 y 1-3 tienen anchos de banda de 5 Mb, un delay de 20 ms. y usan cola *DropTail*. El enlace 1-2 dispone de un ancho de banda de 0,3 Mb, con un delay de 100 ms. y también cola *DropTail*.

Para este escenario usamos los siguientes agentes:

- tcp (Agent/TCP) que actúa como el emisor de la conexión y se asocia al nodo 0.
- sink (Agent/TCP) actúa como receptor de la conexión enviando ACKs al agente tcp. Está asociado al nodo 3.
- ftp (Application/FTP) que es el que realmente genera el tráfico. Está asociado al agente tcp.

Hemos simulado el escenario (mostrado en la *Figura 8.8*) para intentar estudiar la forma en que la red adapta el ancho de banda del emisor modificando los paquetes descartados. Hemos realizado un total de cinco simulaciones del escenario variando algunos de los parámetros según lo siguiente:

- Tres simulaciones se han usado para variar el tamaño de la cola del nodo 2. De este modo se obtienen tres gráficos con la longitud de la cola a 16, 8 y 4 paquetes.
- Otras dos simulaciones en las que se mantiene fijo el tamaño de la cola a 8 paquetes, pero variando el ancho de banda del enlace 1-2 en primer lugar a 0,2 Mbps y después a 0,5 Mbps.

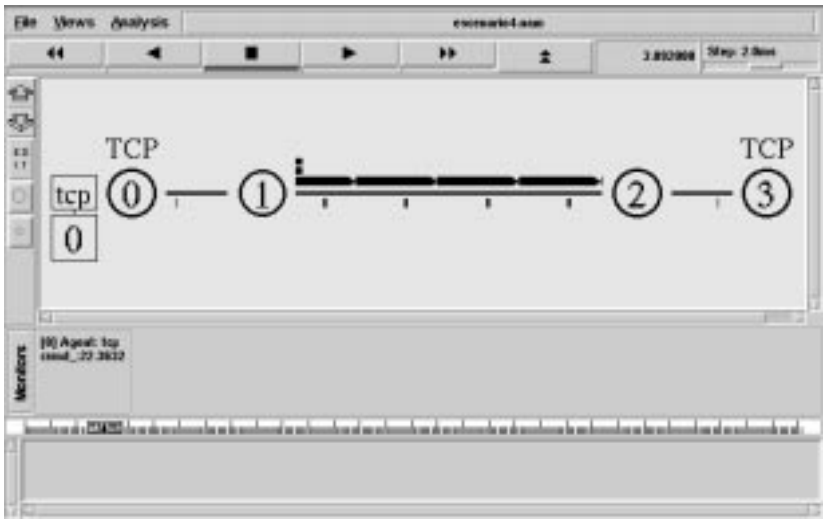

Figura 8.8. *Escenario NS para el estudio del comportamiento del tamaño de las colas y del ancho de banda*[2]

En todas las simulaciones el objetivo es analizar la evolución de la ventana de congestión en el agente emisor TCP comentado anteriormente. A continuación se comentan los resultados obtenidos en las cinco simulaciones realizadas con NS.

De la simulación del anterior escenario hemos obtenido con NS la *Figura 8.9*, donde puede observarse la evolución de CWND dependiendo del tamaño de la cola que se emplee en el router. Si la longitud de la cola es suficientemente grande (L=16) la red no necesita descartar ningún paquete en los 30 segundos estudiados, por lo que la ventana CWND evoluciona con un crecimiento lineal a partir del segundo 2, en que se alcanza el valor del umbral SSHTRESH y empieza a actuar el mecanismo *Congestión Avoidance*.

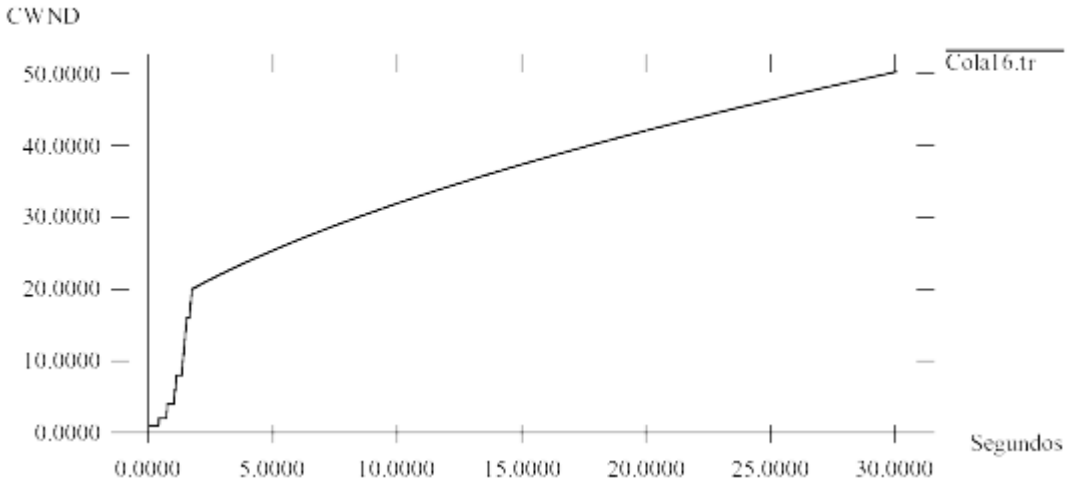

Figura 8.9. *Evolución de la ventana CWND con una cola de 16 paquetes*

[2] Puede observarse el crecimiento de la cola del nodo 1, así como los segmentos y sus correspondientes ACK

Sin embargo, observando la *Figura 8.10* donde se emplea una cola de 8 paquetes podemos comprobar cómo la red obliga al emisor TCP a ajustar su ventana de congestión a la nueva capacidad de la cola. Como puede observarse en la *Figura 8.10* y en la *Figura 8.11*, se ha seguido disminuyendo el tamaño de la cola del buffer de forma progresiva (L=8 y L=4), lo que acaba provocando el descarte de paquetes y acarrea el descenso del throughput en los enlaces. Observemos cómo a medida que decrementamos el tamaño de las colas, también se produce el descenso de la ventana CWND representada en el eje Y.

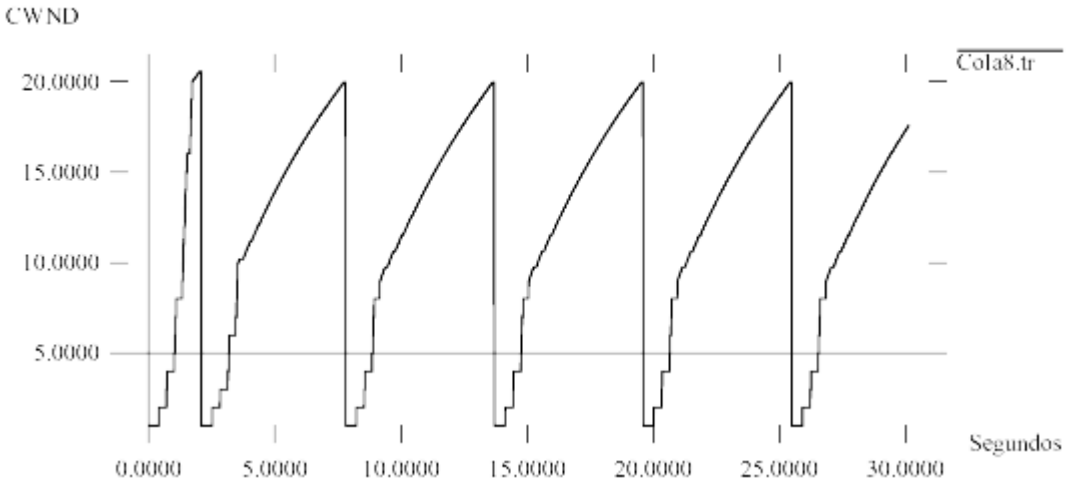

Figura 8.10. *Evolución de la ventana CWND con una cola de 8 paquetes*

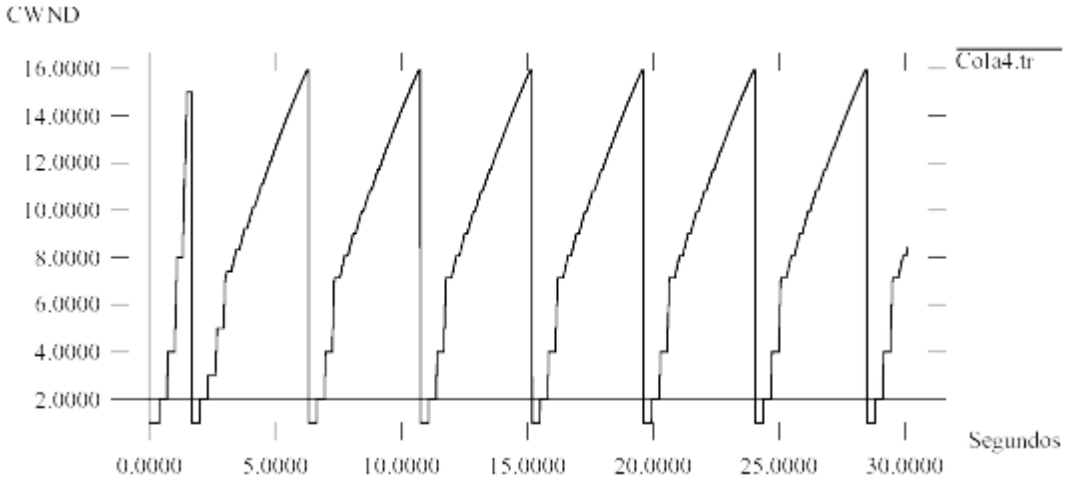

Figura 8.11*. Evolución de la ventana CWND con una cola de 4 paquetes*

En las tres simulaciones presentadas anteriormente no se han tenido en cuenta los paquetes almacenados "*in flight*"[3], porque hemos considerado que la capacidad de almacenamiento de la red está determinada solamente por la capacidad de almacenamiento del router. Sin embargo, aclaramos que la capacidad de almacenamiento de la red es la suma de la longitud de la cola del router más los paquetes almacenados "*in flight*" en el enlace, lo que está determinado por el ancho de banda de cada enlace. Las *Figuras 8.12* y *8.13* demuestran esta situación, en las que podemos observar cómo al disminuir el ancho de banda de un enlace de 0,5 Mb a 0,2 Mb, disminuye también la capacidad de almacenamiento de la red y, por tanto, también desciende el throughput en la misma. Podemos observar cómo al incrementar el ancho de banda de la red a

[3] La capacidad de almacenamiento real de la red es la suma de los paquetes contenidos en las colas de los routers más todos aquellos que en cada momento se encuentran viajando por los enlaces entre nodos.

0,5 Mbps en la *Figura 8.13* se consigue evitar que, en el tiempo de simulación de 30 segundos, la ventana de congestión de TCP no sea puesta a 1 en cinco ocasiones como ocurre en la *Figura 8.12*. Al incrementar la capacidad del enlace, la ventana CWND no es reducida a 1, sino que crece linealmente según el control de *Congestion Avoidance*.

Destacamos que si se aumenta el tiempo de simulación de la *Figura 8.12*, la congestión acabará apareciendo, por las propias características de TCP[4]. Lo que se consigue aumentando el ancho de banda del enlace es retrasar el momento de aparición de las congestiones, mejorando el goodput por tanto.

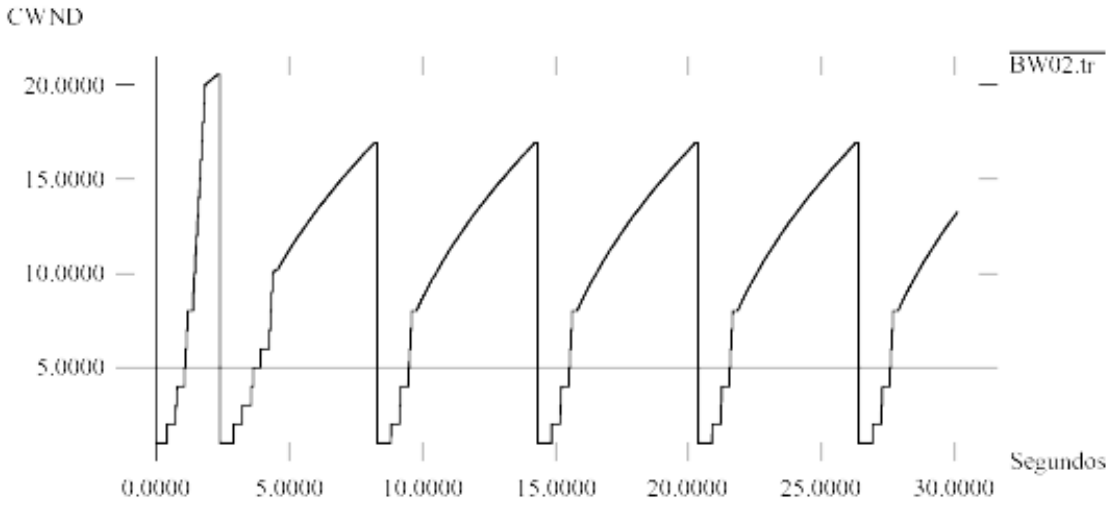

***Figura 8.12**. Evolución de CWND sobre un enlace de 0,2 Mbps*

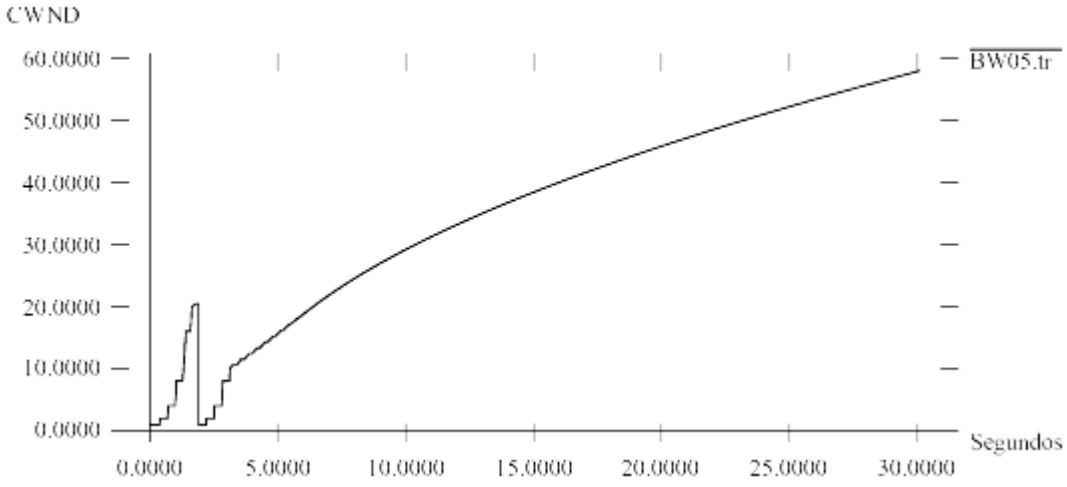

***Figura 8.13**. Evolución de CWND sobre un enlace de 0,5 Mbps*

Si deseamos considerar el efecto de *N* fuentes TCP compartiendo un enlace, debemos de partir de la posibilidad que ese enlace pueda convertirse en un posible cuello de botella determinado por el tamaño de la cola del router que podemos suponer con valor *B*. Los tamaños de las colas TCP deben sumar el tamaño *B*,

[4] Aunque una vez iniciado el algoritmo *Control Avoidance* el crecimiento de la ventana CWND es lineal, llegará un momento en que se supera la capacidad de algún enlace de la red, o bien la capacidad de alguno de los routers que la forman, por lo que CWND será reducido a 1 en el momento que aparezca la congestión.

de forma que *W=B/N*. Sustituyendo estos valores en la ecuación (9) podemos calcular una nueva aproximación a la predicción de probabilidad de pérdida de paquetes:

$$P = 0{,}75\frac{N^2}{B^2} \tag{10}$$

Puede entenderse por la fórmula anterior que cargas elevadas de la red acabarán generando pérdidas elevadas de paquetes, donde además influye directamente el número de fuentes de tráfico que intervienen compartiendo el enlace y, por tanto, también el buffer de los routers. Además, la posibilidad de gestión de la carga de tráfico depende directamente de la capacidad de paquetes del propio buffer que se comporta como un punto de congestión para el tráfico.

Si usamos *N* fuentes, la capacidad de almacenamiento de la red será entonces la suma de todas las fuentes TCP compartiendo el ancho de banda *RTT* veces [7,11]. Por tanto, si las fuentes TCP tienen igual *RTT*, la capacidad de almacenamiento del enlace es el ancho de banda *RTT* veces. De este modo, en la ecuación (10) el valor de *B* es la capacidad de almacenamiento del enlace más los buffers de las colas de los routers.

Por último, podemos entender cómo pequeños límites en la longitud de la cola provocan que en la fórmula (10) la variable *B* se convierta en constante y provoque que la velocidad de pérdidas crezca cuando se incrementa el numero de conexiones TCP compitiendo por el enlace. Como la velocidad de pérdida crece, cada ventana de TCP se ve reducida y cada fuente TCP aminorará su velocidad de envío. De este modo la velocidad de envío *V* de cada fuente puede ser obtenida de las fórmulas (4) y (8),

$$V = \frac{0{,}866}{RTT\sqrt{P}} \tag{11}$$

Llegados a este punto hemos de destacar que la inclusión de la memoria DMTE en la arquitectura TAP contribuye de forma directa a aumentar el tamaño de buffer y, a la vista de la fórmula (10), a conseguir disminuir la probabilidad de pérdida *P* que es inversamente proporcional al tamaño del buffer de las colas en los routers. Sin embargo, nuestra aportación va más allá de la simple agregación de memoria, ya que la ganacia más importante la obtenemos en la posibilidad de realizar las recuperaciones de forma local al router (conmutador) congestionado consiguiendo un valor de *RTT* más pequeño (como se explica en la sección 8.4). TAP consigue, por tanto, disminuir la probabilidad de pérdida *P* y también el *RTT*. Podemos ver que estas dos variables afectan a la fórmula (11) que muestra cómo valores bajos de *RTT* y de *P* permiten alcanzar mayores velocidades de envío en las fuentes de tráfico.

8.3. TCP SOBRE ATM

En lo relativo a las investigaciones sobre la evaluación del rendimiento de TCP sobre ATM éstas se pueden dividir en tres grandes grupos [6]: 1) las que se fijan en el dinamismo de TCP; 2) las que atienden al comportamiento de ATM; y 3) las que prestan atención a la interacción entre las ventanas de TCP y los mecanismos de control de congestión de la capa ATM. Aunque la evaluación del rendimiento de TCP sobre ATM ha sido fuente de diversas investigaciones, las propuestas resuelven sólo problemas particulares como es la fragmentación de TCP, los requerimientos de buffers, la interacción entre los esquemas de congestión de TCP y ATM, y la degradación de TCP. En cierto modo se echan en falta propuestas generales que se enfrenten a todas o varias de estas problemáticas y en esta línea se dirigen nuestras investigaciones, aportando un SMA que busca la optimización del goodput con un adecuado tratamiento de las colas de entrada, y con una cuidada política de gestión de buffer mediante la delegación de actividades concretas en los agentes que forman parte del SMA.

La mayor parte de aplicaciones de datos no son capaces de predecir sus propias necesidades de ancho de banda, por lo que se necesita de algún servicio que permita a todos los usuarios activos de la red compartir dinámicamente el ancho de banda disponible. Sabemos que en el caso de ATM las CoS ABR y UBR son la propuesta estándar para soportar el tráfico de datos. La referencia [2] presenta el estudio de congestiones de redes TCP sobre ATM comprobando cómo el throughput de TCP cae también cuando se comienzan a descartar células en los conmutadores ATM. El bajo throughput conseguido se debe al desaprovechamiento del ancho de banda en los enlaces congestionados que transmiten células de paquetes corrompidos, es decir, paquetes en los cuales se ha tirado alguna de sus células[5]. Otras investigaciones [12] han demostrado que

[5] En realidad este trabajo estudia el efecto de la fragmentación de los segmentos TCP.

TCP sobre UBR con EPD experimenta una apreciable degradación en el funcionamiento en cuanto a la justicia, requiriendo además de un tamaño de buffer relativamente grande, incluso con pocas conexiones. Sin embargo, cuando se emplea la CoS ABR con esquemas de realimentación de velocidad explícita ofrecen a TCP mejor comportamiento en cuanto a justicia y aprovechamiento de los enlaces y, todo ello, con tamaños de buffer menores que con UBR.

La literatura [13] describe también otras formas de evitar la degradación del througput de fuentes TCP sobre UBR. Para ello lo que se hace es desactivar los descartes de células ATM durante un periodo de tiempo en el que se están produciendo congestiones. De este modo se evitan los timeouts de TCP que son la principal causa de descenso del throughput de TCP, y se acortan los periodos de congestión evitando el gran retardo experimentado con el algoritmo de retransmisión rápida de TCP antes de que el emisor reciba los ACK duplicados.

Uno de los principales problemas que experimentan las fuentes de tráfico TCP cuyos segmentos son transferidos sobre ATM es que, al usarse como indicación de congestión la propia pérdida de paquetes, esto provoca que la gestión de las congestiones se realice cuando ya es demasiado tarde. Es decir, cuando el buffer se llena, el conmutador descarta los paquetes, el receptor detecta la pérdida y éste avisa al emisor que acaba reaccionando con la retransmisión de los paquetes perdidos. Se impone por tanto la necesidad de aportar mecanismos más ágiles para la solución de las congestiones del tráfico TCP que es transferido sobre tecnología ATM.

Nos encontramos con la característica que TCP es un protocolo orientado a conexión y fiable (confirmación de segmentos entregados), mientras que ATM es también orientado a conexión pero no ofrece de forma estándar ningún mecanismo de confirmación de células entregadas. Puede hablarse así de comportamientos independientes entre ambas tecnologías. Además, cuando TCP funciona sobre ATM, el control de la red es más complejo por requerirse un mecanismo de control de congestión diferente en cada una de las capas. Existen dos diferencias básicas entre los esquemas de control de congestión de TCP y la CoS ABR de ATM: 1) El mecanismo de realimentación de ABR con células RM controla la velocidad de transmisión de las células desde el emisor (control de velocidad), mientras el mecanismo de realimentación de TCP controla el tamaño de una ventana como hemos visto en el apartado anterior (control de créditos) y 2) el mecanismo de realimentación de ABR puede ser realizado por conmutadores intermedios de la red, o por el extremo receptor del tráfico, mientras en TCP el mecanismo de realimentación es realizado sólo por el nodo destino mediante ACK extremo-a-extremo que es lo que aporta la fiabilidad a TCP.

En las fuentes TCP el tráfico máximo es controlado por la ventana CWND tal como hemos demostrado en el apartado anterior. Sin embargo, en el caso de la CoS ABR de ATM el tráfico es controlado por diversos parámetros como MCR, PCR y ACR. Son también aspectos clave para TCP sobre ATM los mecanismos de gestión de tráfico usados en los nodos extremos de TCP, en los nodos extremos de ATM y en los conmutadores de la red para, entre todos ellos, aportar el adecuado goodput para reducir el retardo causado por las retransmisiones. El retardo de procesamiento de los paquetes TCP es un periodo de tiempo aleatorio que modela la media del retardo de procesamiento de paquetes con una cierta variación de retardo. Esto se aplica, tanto a paquetes de datos, como a los de confirmación entre emisores y receptores

A la vista de todas estas características diferenciadoras y, dado que ATM es siempre un protocolo situado por debajo del protocolo de la capa de transporte TCP, se requieren soluciones para resolver los problemas de rendimiento provocados por la integración de ambas tecnologías. Estas soluciones parece lógico que estén en la línea de realizar cambios en los conmutadores ATM dentro de la red, o bien en la nueva implementación de extensiones para TCP, o también en la propuesta de protocolos especializados para los nodos que están en los límites de la red ATM con la red TCP. Destacamos que en nuestro caso TAP se enfrenta a estos problemas actuando dentro de la misma red con mecanismos hardware (conmutadores AcTMs) y también software (SMA con protocolo TAP), lo que configura toda la arquitectura TAP objeto de esta tesis.

8.4. BENEFICIOS APORTADOS POR TAP

Como ya sabemos, en nuestro caso nos basamos en EAAL-5 como extensión de AAL-5 que se diseñó específicamente para la comunicación de datos a través de ATM. En el caso de TCP sobre ATM, los datagramas IP son transmitidos en la zona del campo de datos (payload) de EAAL-5, tal como podemos intuir en la *Figura 8.14* que presenta las pilas de protocolos de una fuente emisora y otra receptora basada en TCP sobre ATM con UBR.

Por otro lado, también hemos comentado ya que nuestras propuestas van dirigidas al tráfico UBR y ABR que puede servir perfectamente para el soporte del tráfico TCP. La CoS ABR soporta aplicaciones que generan y gestionan tráfico de datos. ABR tiene la capacidad de reducir su velocidad de envío de tráfico si se

producen congestiones en la red. Por esto, podemos decir que la CoS es en realidad un mecanismo evitador de congestiones (Congestion Avoidance), y el ATM Forum ha adaptado un control de flujo para ABR conocido como *Control de Congestión basado en velocidad.* Sabemos ya que este mecanismo está basado en realimentación en la propia red usando células especiales RM. Si consideramos el esquema de velocidad explícita (ER), la fuente de tráfico envía al destinatario por el VPI/VCI una célula cada N[6] células, o bien se genera una célula RM cada T[7] unidades de tiempo. Cada célula RM contiene tres campos que aportan la realimentación a la fuente: el bit de Indicación de Congestión (CI);el bit de No Incremento (NI) y el campo de Velocidad Explícita (ER). La fuente de tráfico fija el valor del campo ER de la célula RM indicando la velocidad a la que desea enviar datos y a continuación transmite el célula FRM (Forward RM). Cuando la célula FRM es recibida por el destinatario, ésta es devuelta al emisor como célula BRM (Backward RM). Cualquiera de los campos CI, NI y ER pueden ser cambiados por los conmutadores ATM que procesan el VC del destinatario, antes de que la BRM llegue a la fuente que usará esta información para conocer el estado de la red, información que usará para ajustar su tasa de envío si se detectan congestiones. Se han propuesto varios esquemas basados en velocidad como EPRCA (Enhanced Proportional Rate Control Algorithm) o el esquema elegido por el ATM Forum ERICA (Explicit Rate Indication Congestion Avoidance).

Antes de concluir estos breves comentarios sobre el control de congestión de ATM hemos de destacar que éste es un importante aspecto para esta tecnología, por lo que se ha puesto especial interés en aspectos como: la escalabilidad, la justicia, la robustez y la facilidad de implementación del control de congestión. Aspectos éstos que hemos procurado satisfacer en nuestra propuesta.

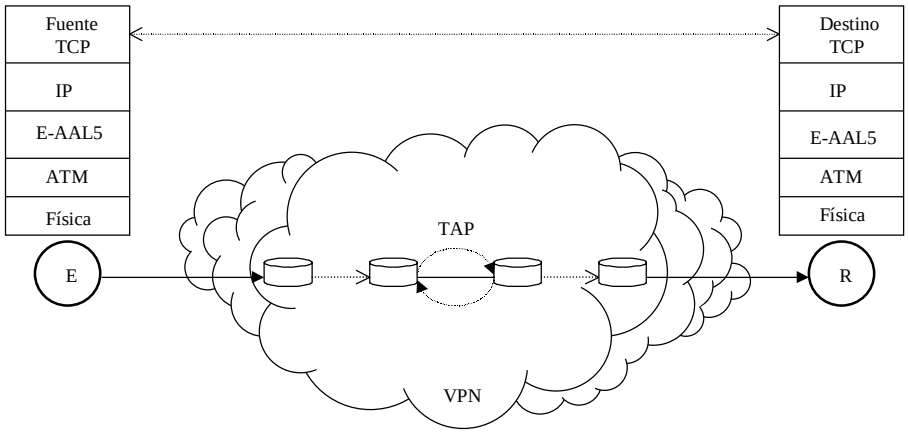

Figura 8.14. *TCP sobre ATM con TAP*

El efecto del tiempo *RTT* con respecto a la velocidad de transmisión puede comprenderse claramente observando la *Tabla 8.1* en la que se presentan diferentes portadoras ópticas, tanto de la jerarquía digital síncrona (SDH), como de la jerarquía de multiplexación SONET. La *Tabla 8.1* presenta los tiempos necesarios para la transmisión de ficheros de diversos tamaños con respecto a enlaces con diferentes velocidades de transmisión.

Nuestra intención es justificar la necesidad de mecanismos que sirvan para evitar los costes de los tiempos *RTT* en el caso de congestiones. Para ello supongamos que la VPN de la *Figura 8.14* tiene 2.000 Kms. de longitud extremo-a-extremo. Supongamos también que empleamos fibra óptica como medio físico de transmisión, en la que sabemos que se obtienen velocidades de propagación aproximadas a 2/3 la velocidad de la luz en el vacío, es decir, unos 200.000 Kms/s. En este escenario un bit tardará en recorrer la red 10 ms., por lo que, en el mejor de los casos (libre de errores y de congestiones), el *RTT* será de 20 ms.

Si estudiamos la *Tabla 8.1* con detenimiento podemos destacar que en esa misma red la transferencia de un fichero de 512 Kbytes a 155,52 Mbps supondrá un tiempo total de 3,3 ms. para llegar desde un extremo a otro de la red. Suponiendo que el fichero llegase en un solo paquete, y sin experimentar ningún problema en la transmisión, la transferencia tendrá un coste total de 23,3 ms., de los cuales 20 ms. pertenecen al coste de transferencia y al de confirmación de llegada del paquete desde el receptor, suponiendo que se trata de una fuente TCP. Por tanto, el enlace realmente está en estado de inactividad durante 20 ms., lo que supone un total del 86 % del tiempo total.

[6] El valor por defecto de N es 32. Por tanto, se genera de forma fija una célula RM de realimentación cada 32 células de datos.

[7] El valor por defecto de T es de 100 ms. Por tanto, se genera de forma constante una célula RM de realimentación cada 100 ms.

TABLA 8.1
RELACIÓN DE TIEMPOS DE TRANSMISIÓN RESPECTO A VELOCIDADES DE TRANSMISIÓN

SONET	SDH	Velocidad de transmisión	1 Mbytes	512 Kbytes	256 Kbytes	20 Kbytes
OC-1	-	51,84 Mbps	161,8 ms.	10,1 ms.	5 ms.	395 µs.
OC-3	**STM-1**	155,52 Mbps	53,9 ms.	3,3 ms.	1,6 ms.	131 µs.
OC-12	**STM-4**	622,08 Mbps	13,4 ms.	842 µs.	421 µs.	32,9 µs.
OC-24	**STM-8**	1.244,16 Mbps	6,7 ms.	421 µs.	210 µs.	16,4 µs.

La situación descrita se ve acrecentada a medida que usamos velocidades de transmisión más elevadas. Si consideramos usar un enlace OC-12 de 622,08 Mbps para la transferencia del mismo fichero de 512 Kbytes la diferencia entre el tiempo de transferencia total del fichero (0,842 ms.) y el *RTT* (20 ms.) es aún mayor que en el caso anterior, por lo que el enlace está desaprovechado el 96,3% del tiempo.

Los estudios relacionados con las transferencias realizadas a través de las redes demuestran que la mayor parte de ellas consisten en ficheros del orden de pocas decenas de Kbytes. Así, si consideramos un escenario similar al presentado en los párrafos anteriores, pero con la transferencia de un fichero de 20 Kbytes podemos comprobar a la vista de la *Tabla 8.1* que si se emplea nuevamente OC-12, el tiempo total de transferencia y de confirmación de la misma supondrá 20,032 ms. En esta situación la fuente del tráfico estará enviando tráfico durante 32,9 µs. y el resto de tiempo corresponde al *RTT*. Por esto la fuente se mantendrá en actividad el 0,16% del tiempo y el enlace quedará desaprovechado el 99,84 % del tiempo total.

Podemos comprobar que en el ejemplo propuesto en la *Figura 8.15*, a medida que se incrementan las velocidades de los enlaces, o se decrementa el tamaño de los ficheros transmitidos, el *RTT* se aproxima asintóticamente a 20 ms., mientras el porcentaje de utilización del ancho de banda del VPI/VCI usado para las transferencias se aproxima a 0.

En el escenario descrito hemos supuesto una red ideal en la que no se produce ningún problema debido a congestiones o bits erróneos. Fácilmente podemos entender que cualquier tipo de retransmisión debida a estos problemas acaba agravando aún más el problema descrito, ya que se multiplica el tiempo *RTT* mientras el tiempo de actividad de la fuente se mueve en valores cercanos a 0.

La motivación general de nuestra tesis se encuentra por tanto en encontrar soluciones para aliviar este negativo problema de retransmisiones extremo-a-extremo por lo que TAP se enfrenta a resolver las retransmisiones de forma local a donde se producen para evitar el coste del tiempo *RTT* total de la red y emplear sólo el *rtt* del enlace donde se produce la congestión. A la vez, aprovechamos los tiempos de inactividad en que se encuentran las fuentes de tráfico cuando se usan enlaces rápidos. Es decir, empleamos los tiempos de inactividad de los enlaces para realizar las retransmisiones punto-a-punto en lugar de hacerlas extremo-a-extremo.

De forma más intuitiva podemos ver también la situación descrita en la *Figura 8.15*, donde tenemos una red con 6 enlaces en la que cada uno de ellos experimenta igual retardo *d=10 ms*. En una situación de red ideal, el retardo total de la red es de *D=60 ms*, por lo que tenemos un *RTT=120 ms.* entre los dos extremos de la comunicación. Nuestro objetivo es situar el protocolo TAP entre dos enlaces de la red para conseguir amortiguar el efecto negativo de las congestiones producidas en el conmutador 3. Si suponemos que, tanto el conmutador 2 como el 3 soportan la arquitectura TAP, cuando se produzca congestión de un paquete en el conmutador 3, su retransmisión no tendrá un coste de 150 ms, sino que en realidad el coste será de 20 ms. que corresponde el retardo debido al *rtt* local del enlace que ha experimentado la congestión

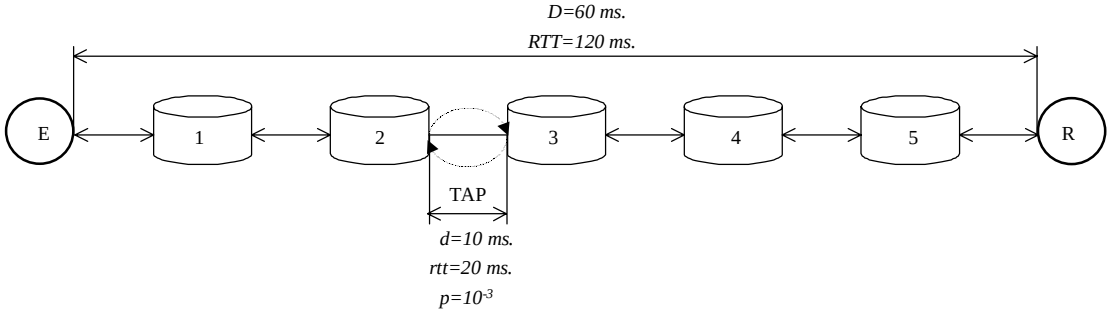

Figura 8.15. *Retransmisión local al conmutador congestionado*

Ante la *Figura 8.15* podemos retomar las fórmulas (10) y (11) donde se expresan las relaciones entre *RTT*, *B* (tamaño del buffer), *P* (probabilidad de pérdida) y *V* (velocidad de envío de las fuentes). Destacamos que en el caso ideal en que no se produzcan pérdidas de segmentos TCP en la red, la memoria DMTE de TAP aporta mayor tamaño de buffer lo que colabora a evitar las pérdidas. Por otro lado, si aparecen pérdidas, TAP las recupera en el punto donde se producen por lo que se reduce el *RTT* extremo-a-extremo al *rtt* punto-a-punto.

8.5. CONCLUSIONES

En este capítulo hemos centrado algunas de las motivaciones generales de esta tesis relacionándolas con las actuales necesidades que tiene la tecnología ATM para soportar la gran cantidad de aplicaciones (no nativas ATM) de datos existentes en las que se usa el protocolo TCP de la capa de transporte. Aunque la arquitectura TAP está pensada para soportar el tráfico ATM nativo, también puede aportar sus características intrínsecas a las transferencias de fuentes basadas en TCP.

En los protocolos de la capa de transporte como TCP sobre ATM un paquete es descartado por la red cuando se pierde una o varias células del paquete, y el nodo destino solicita la retransmisión completa del paquete corrompido o perdido. Hemos demostrado mediante simulaciones la degradación que experimenta el throughput de TCP viendo cómo éste cae logarítmicamente a medida que aumenta la probabilidad de pérdida de células ATM. Con TAP se realizan las retransmisiones de forma local por lo que se consigue decrementar la probabilidad de pérdida con el consiguiente efecto sobre el throughput de TCP que evita los retardos debidos al RTT extremo-a-extremo. Se consigue así maximizar el goodput con una mínima pérdida de paquetes en el nivel TCP y con un retardo mínimo de células en el nivel ATM.

REFERENCIAS

[1] W. Richard Stevens, "TCP/IP Illustrated, Volume 1," *Addison-Wesley Professional Computing Series*, (1994).

[2] Romanow, A. and Floyd, S., "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, pp. 633-641, (1995).

[3] UCB/LBL/VINT Network Simulator - ns, *http://www-mash.cs.berkeley.edu/ns/*

[4] V. Jacobson, "Congestion Avoidance and Control," *ACM Computer Communications Review, proceedings of SIGCOMM'88*, pp. 314-329 (Aug. 1988).

[5] L.S. Brakmo, S. W. A'Malley, and L.L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Proceedings of SIGCOMM'94* (1994).

[6] K. Djemame, and M. Kara, "Proposals for a Coherent Approach to Cooperation between TCP and ATM Congestion Control Algorithms," *Proceedings UKPEW'99*, pp. 273-284, *http://www.cs.bris.ac.uk/Events/UKPEW1999/proceedings/* (1999).

[7] Robert Morris, "Scalable TCP Congestion Control," *Proceedings IEEE INFOCOM'2000*, pp.1176-1183, (2000).

[8] András Veres, and Miklós Boda, "The Chaotic Nature of TCP Congestion Control," *Proceedings IEEE INFOCOM'2000*, pp. 1715-1723, (2000).

[9] Amit Aggarwal, Stefan Savageand Thomas Anderson, "Understanding the performance of TCP Pacing," *Proceedings IEEE INFOCOM'2000*, pp. 1157-1165, (2000).

[10] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, "The Macroscopic behavior of the TCP Congestion Avoidance Algorithm," *Computer Communications Review of ACM SIGCOMM*, vol 27, n. 3, (1997)

[11] Sally Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *Computer Communications Review*, vol 21, n. 5 (1995).

[12] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Ikeda, C., and Suzuki, H., "A simulation study of TCP performance in ATM networks with ABR and UBR services," *Proceedings IEEE INFOCOM'96*, pp. 1269-1276, (1996).

[13] Shunsaku Nagata, Naotaka Morita, Hiromi Noguchi, and Kou Miyake, "An analysis of the impact of suspending cell discarding in TCP-over-ATM," *Proccedings IEEE INFOCOM'2000*, pp. 1147-1156, (2000).

CAPÍTULO 9

LIMITACIONES DE ATM FRENTE A LA GOS Y PROPUESTAS DE SOLUCIÓN

9.1. INTRODUCCIÓN

En el capítulo anterior se han destacado las motivaciones generales de nuestras investigaciones, las cuales pueden aportar atractivas ventajas, no sólo al tráfico nativo ATM, sino también a las aplicaciones que generan en la actualidad la mayor parte del tráfico basado en TCP/IP.

También en capítulos previos de la *Parte I* se han introducido la mayor parte de investigaciones relacionadas con nuestras aportaciones donde hemos identificado de forma general las limitaciones que experimenta la tecnología actualmente. En este capítulo vamos a presentar agrupadas y ampliadas todas estas limitaciones ya identificadas que impiden ofrecer a través de ATM el concepto de la GoS que proponemos como derivada de las parámetros generales de QoS ya analizados. Destacamos que para ofrecer la GoS al conjunto de conexiones privilegiadas que lo requieran sobre la VPN que proponemos se han de buscar soluciones para todas estas limitaciones.

Hemos partido en nuestras propuestas de la imposibilidad para aportar fiabilidad total a ATM sin afectar al *goodput* de la red. Por esto, nuestra intención se centra en garantizar servicio a aquellas conexiones que lo requieran a través del protocolo TAP. TAP aprovecha la fiabilidad ofrecida por el campo HEC de las cabeceras, así como el CRC de las colas de las PDU. Sin embargo, el protocolo aporta su propio mecanismo para solventar las congestiones para las que ATM no dispone de ninguna solución estándar que no se base en retransmisiones extremo-a-extremo que acaben degradando el comportamiento de la red.

En general, las soluciones a los problemas de congestiones se basan en equipar a los nodos intermedios de la red con mecanismos que permitan solventar estas situaciones de forma local sin implicar, ni al resto de la red, ni tampoco a las fuentes emisoras. Para conseguir esto en ATM necesitamos identificar las PDU de cada una de las fuentes, por lo que aprovechamos las posibilidades que nos aportan los estándares para proponer nuestro propio mecanismo de numeración de los paquetes de datos generados por los emisores.

La tecnología ATM no emplea ningún mecanismo de confirmación de células entregadas para conseguir el mejor *throughput* posible que se vería afectado si se confirmasen las células de forma individual. Sin embargo, pueden usarse muy diversos mecanismos de solicitud de retransmisión de células perdidas, siempre agrupadas en PDU. Este mecanismo de *feedback* puede acabar provocando el también indeseable problema de la implosión sobre la fuente emisora de tráfico. La implosión es indeseable en todo tipo de aplicaciones pero, sobre todo, en aquellas en las que están implicados varios destinatarios en conexiones p-mp ó mp-mp. Por esto, este es otro problema que atrae especialmente nuestra atención por el efecto que puede generar sobre la GoS de la fuente, de los destinos y de toda la red en general.

Por otro lado, aparecen otros fenómenos indeseables que provocan que las conexiones ATM no dispongan de servicio garantizado. Este tipo de problemas está también relacionado con las impredecibles congestiones que pueden producirse en los conmutadores. Uno de estos problemas es la fragmentación de las PDU que se ven afectadas por una congestión y que son descartadas parcialmente por no tener cabida en el buffer de los conmutadores. Éstos no son capaces de tratar las PDU como unidades de transferencia en la

conmutación, por lo que las PDU pueden continuar su camino en dirección al destino fragmentadas cuando sabemos "a priori" que serán detectadas como corruptas por haber perdido parte de sus células.

La multiplexación en el mismo buffer de las células que provienen de varias fuentes provocan el segundo problema que es el de la mezcla (*interleaving*) en el mismo puerto de salida de células alternas de diferentes conexiones que tiene salida por el mismo enlace, aunque no vayan dirigidas a los mismos destinatarios.

Podemos decir que la tecnología ATM basa la mayor parte de su rendimiento en dos exigencias básicas como son la máxima velocidad de conmutación y la mínima complejidad en los conmutadores. Pues bien, la arquitectura TAP se apoya en la posibilidad de equipar a la propia red con elementos hardware y mecanismos software que podrían parecer contradictorios con las dos exigencias citadas. Por esto justificamos que conseguimos nuestros objetivos sin incrementar el retardo ni la complejidad de la red.

Todos estos aspectos serán desarrollados en los siguientes apartados de este capítulo que tendrá continuación en la *Parte III* con la descripción detallada de TAP para mejorar la tecnología que permita disponer de GoS en las conexiones privilegiadas que requieran de este servicio. En cierto modo, este capítulo pretende ser un compendio de las propuestas que ya hemos ido adelantando en capítulos precedentes, y que serán expuestas con más detalle a continuación.

9.2. TRANSFERENCIAS ATM NO FIABLES

En capítulos anteriores hemos justificado las dificultades de la tecnología ATM para aportar fiabilidad total en sus transferencias. Hemos identificado ya las diversas propuestas existentes para garantizar en la medida de lo posible el control de errores. De este modo sabemos que el campo HEC de las cabeceras de las células está pensado para detectar los posibles errores sólo en las cabeceras. El campo de datos de las células no tiene control de errores en células independientes pero, sin embargo, se dispone del campo CRC-32 en las PDU de AAL-5 para la detección de errores en una unidad de transferencia de tamaño mayor que las células. Sabemos también que el mecanismo FEC puede aportar la mayor seguridad posible en cuanto a la aparición de errores se refiere pagando un importante coste en *overhead* debido al código redundante que introduce. Sin embargo, hemos partido también de la imposibilidad de disponer de fiabilidad total sin afectar al buen rendimiento de la red. Aunque es factible la detección de errores, tanto en cabeceras como en datos, lo que no podemos garantizar es que las células encargadas de las retransmisiones puedan perderse en la red por congestión en los conmutadores o por fallos de conmutación provocados por errores en las cabeceras de esas células. De este modo, la red asume que pueden producirse pérdidas, de forma que la resolución de las mismas quedan delegadas en protocolos de capas superiores.

Por otro lado, en el capítulo anterior hemos visto que protocolos de capas superiores como TCP pueden tener un comportamiento bastante caótico en cuanto al control de errores y al control de congestión se refiere. No obstante debemos de destacar que en nuestra arquitectura y protocolo TAP asumimos las limitaciones relativas a la fiabilidad, porque nuestro principal objetivo es resolver, no los problemas relacionados con los errores en la transmisión, sino los planteados cuando se producen congestiones en los conmutadores, que son más graves y frecuentes. Por esto, TAP asume que el control de errores va a regirse por las propuestas estándares de la propia tecnología basadas en HEC para las células y en CRC para las PDU.

9.3. NÚMEROS DE SECUENCIA INEXISTENTES

La mayor parte de protocolos de comunicaciones de las capas de Enlace y de Transporte se enfrentan a los problemas de control de flujo y de errores contando con la posibilidad de identificar los paquetes que procesan mediante un identificador que es el que emplean para la detección de paquetes perdidos (cuando falla la secuencia) y para la solicitud de retransmisiones (generalmente e-e). Sin embargo, la tecnología ATM no dispone de números de secuencia ni en las células (para evitar el *overhead*), ni en unidades de transferencia mayores de la capa AAL (como pueden ser las PDU de AAL-5). Esta ausencia de números de secuencia en células y PDU impide la identificación de las unidades de transferencia que se pierden cuando la red experimenta congestiones. Esto obliga a que la detección de datos perdidos sea delegada a los protocolos de capas superiores que, como sabemos, repercute en la eficiencia de la red por la necesidad de realizar las retransmisiones e-e.

Como ya se ha comentado en capítulos precedentes, uno de los componentes de la arquitectura TAP es una extensión de la capa AAL-5 cuyo principal objetivo es precisamente el de aportar a las PDU de AAL-5 una numeración de la secuencia de paquetes que el nodo emisor está generando. El objetivo de estos números de secuencia es la posibilidad de poder identificar las PDU que puedan perderse por congestión y disponer

así de un mecanismo que nos sirva para su retransmisión de forma local, en lugar de delegar esta función en el nodo destinatario de la conexión.

AAL-5 fue propuesto [1] para reducir las sobrecargas de cabeceras introducidas por AAL3/4. La *Figura 9.1* compara el formato de CPCS-PDU de AAL-5 nativo con EAAL-5 con todos sus campos. Como podemos ver, la cola de las PDU tiene cuatro campos. El campo *User-to-User indication* (CPCS-UU) es usado para la transferencia de información CPCS de usuario a usuario. El octeto *Common Part Indication* (CPI) se usa para ajustar la cola CPCS-PDU a 64 bits. Cada uno de los campos de AAL-5 han sido descritos en el apartado 3.2 del *Capítulo 3*. No obstante, hemos de recordar que la Rec. I.363.5 especifica que los dos campos que aprovechamos, cuyo valor habitual es 0, tienen asignadas las siguientes funciones:

- CPCS-UU (Common Part Convergence Sublayer User-to-User): este campo de un byte es transferido de forma transparente entre usuarios CPCS de la capa AAL-5.
- CPI (Common Part Indicator): se usa principalmente para completar o rellenar a 64 bits la cola de las PDU. Cuando se usa como relleno este campo lleva los 8 bits siempre a 0. El único valor legal en la actualidad es 0 en sus 8 bits para indicar que las PDU contienen datos de usuario. Se especifica que otras codificaciones de este campo están en estudio, y por esto hemos decidido usarlo conjuntamente con el campo CPCS-UU para tener una secuencia de PDUs más amplia.

Por tanto, la *Figura 9.1* se corresponde con la *Figura 3.1* en la que hemos decidido sustituir los campos CPCS-UU y CPI del estándar ATM para emplearlos por el campo PDUid de EAAL-5 que es el que se encarga de aporta el número de secuencia a cada una de las PDU. De este modo, el nodo emisor que soporta TAP generará un flujo continuo de datos que va a ser encapsulado en PDU como la de la *Figura 9.1*, de forma que el propio nodo activo se encarga de controlar el número de secuencia a medida que va realizando el encapsulado de datos en PDU. Con los dos octetos que tenemos disponibles para el campo PDUid de EAAL-5 podemos generar una secuencia de hasta 65.535 PDU que aportan una importante "ventana" de retransmisiones en caso que alguna de las PDU se acabe perdiendo por congestión de alguno de los conmutadores de la red. Una vez que se supera un ciclo de la secuencia, ésta es reiniciada nuevamente por el nodo emisor que comienza desde el identificador de PDU número 1. Los estudios realizados, así como las simulaciones, nos permiten afirmar que, aunque pudiese parecer que la secuencia de 65.535 PDUid sea insuficiente, ésta "ventana" es más que suficiente para conseguir nuestros objetivos. Aunque las velocidades de ATM son elevadas, el ciclo de identificadores de PDU permite garantizar que no se van a tener valores de éstos duplicados en la red, ya que, aunque diferentes fuentes puedan elegir como VPI/VCI valores iguales, los índices están compuestos también por el identificador de puerto de entrada y salida en los conmutadores, lo que nos permite disponer de una buena clave primaria de acceso a la memoria DMTE. Este aspecto será justificado detalladamente en el *Capítulo 10*.

El trafico generado por las fuentes (ya sea como nativo ATM o por otros protocolos como TCP) es ensamblado en unidades de PDU (de tamaño variable y como máximo de 65.535 octetos) a los que se añade la cola de la PDU de 8 bytes que puede observarse en la *Figura 9.1*. Entre los campos de la cola se incluye el número de secuencia de cada PDU, el campo de Longitud que permite determinar la PDU (que es de tamaño variable) y, además se calcula el Código Redundante Cíclico sobre la PDU, que será empleado en el nodo destino para detectar errores que serán solventados mediante una retransmisión e-e. El campo de *padding* (relleno) lo empleamos de relleno, como en el estándar, para que la PDU tenga un tamaño final que sea múltiplo de la unidad de conmutación que es la célula ATM. En nuestra arquitectura aprovechamos por tanto el formato estándar de las PDU con la intención de soportar toda la tecnología actual, de forma que en la VPN que se propone no sea necesario cambiar los conmutadores ya existentes para poder disponer de transmisiones garantizadas. De este modo se ofrece la GoS a las transferencias de tráfico ATM nativo tal como vimos en el *Capítulo 2*, y también a fuentes de tráfico como las de TCP vistas en el *Capítulo 8*.

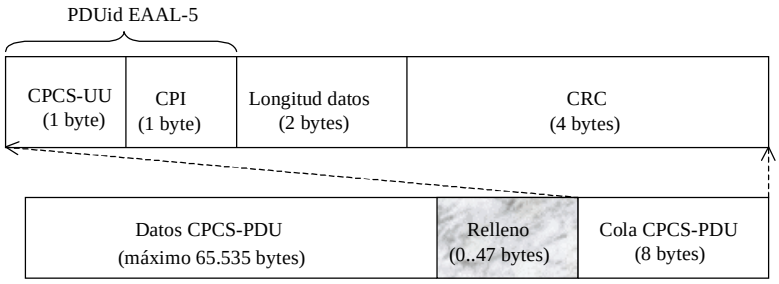

***Figura 9.1**. Formato de CPCS-PDU EAAL-5*

Destacamos que los números de secuencia PDUid son asignados extremo-extremo por los usuarios EAAL-5 para evitar el recálculo de CRC y la modificación de la cola de CPCS-PDU en cada uno de los conmutadores atravesados por el tráfico. El CRC es empleado como en AAL-5 para detectar los bits erróneos en las CPCS-PDU. Es destacable también el hecho que la técnica usada para la obtención de PDUid no introduce ningún *overhead* en las transferencias, ya que empleamos dos campos de la estructura de las PDU estándares que, de otro modo, no se usarían para ninguna otra función específica.

Tal como representa en el *Capítulo 2* la *Figura 2.6*, los conmutadores AcTMs que soportan TAP disponen de la capa EAAL-5 con funciones similares a AAL-5, y además con el control de los números de secuencia, que son también soportados por los conmutadores intermedios de la red sobre la capa ATM. Los conmutadores AcTMs intermedios en realidad no llevan incorporada la capa EAAL-5 sino que únicamente mantienen el control de las PDUid para poder realizar las retransmisiones o para poder atenderlas si es el caso.

Finalmente, en los nodos destino el campo PDUid será empleado por el protocolo EAAL-5 para actuar según los dos tipos de servicio para los que está programado: servicio ordenado *Seq* (secuencial) y servicio connectionless (desordenado). Las simulaciones realizadas nos han permitido comprobar empíricamente que se verifica la idea intuitiva de que cuando aparecen congestiones en la red, aunque TAP es capaz de resolverlas localmente, se produce desorden de PDUs en el destino, por lo que en el caso del servicio ordenado de PDU, EAAL-5 deberá encargarse de ordenarlas (a través del PDUid) antes de ser pasadas a protocolos de capas superiores como el propio TAP, SSCOP, TCP/IP o MPEG. Si se desea servicio no ordenado, las PDUs serán pasadas sin mayor demora a los protocolos de capas superiores que deberán encargarse de la ordenación y de la detección de pérdidas.

9.4. RETRANSMISIONES EXTREMO-EXTREMO

En el *Capítulo 3* tuvimos la ocasión de comentar que nuestra propuesta de GoS está estrechamente relacionada con la técnica ARQ, y concretamente basada en NACK. Esto es así porque una de nuestras aspiraciones es la de obtener el mejor goodput posible de la red. De este modo, con los NACK sólo generamos células en sentido al emisor, cuando se están produciendo pérdidas en la red.

Para implementar NACK usamos células estándares *Resource Management* (RM) propuestas para la clase de servicio ABR (Available Bit Rate), pero sin frecuencia fija[1] y generadas sólo cuando se congestiona un conmutador AcTMs. De este modo evitamos el efecto negativo de las sobrecargas de la red debidas a que un número fijo de células RM desaprovecharían el ancho de banda innecesariamente. Así, en el protocolo TAP sólo se generan células RM cuando el propio protocolo detecta una congestión en uno de los AcTMs, de forma que en ese momento se solicita la retransmisión en el punto en que se produce, y al conmutador previo, generando una célula *Backward* RM. El resto del tiempo no se generan células BRM, a no ser que estemos usando la CoS ABR sobre nuestra arquitectura.

También en esta ocasión hemos querido mantener las propuestas estándares de la tecnología ATM para garantizar la compatibilidad de TAP con las instalaciones existentes, y por esto hemos centrado nuestra atención en sacar partido de las células RM estándares de ABR, pero debidamente ajustadas a nuestras necesidades. Analizando las propuestas del ATM-Forum [2], en cuanto a las RM se refiere, hemos podido comprobar cómo parte de la información que viaja en la RM está desaprovechada y, por lo tanto, nos ha sido de suma utilidad para nuestras intenciones. En realidad, las propuestas estándares han dejado abiertas las posibilidades para que las RM sean empleadas con diversos objetivos y, en nuestro caso, hemos decidido usarlas como un mero mecanismo de comunicación entre conmutadores para conocer la información de las PDU que se han congestionado, en lugar de usarse para conocer el estado de la red.

Las células BRM mantienen la misma estructura que el resto de las células, con la diferencia que se han reservado determinadas combinaciones para funciones muy concretas. Así, existen múltiples y conocidas funciones de los campos de este tipo de células que son presentados en la *Tabla 9.1*. Como podemos observar de forma destacada, los octetos del 22-51 no tienen ninguna función especifica encomendada, por lo que se han reservado para usos futuros. Disponemos de este modo de 29 octetos que empleamos en TAP para identificar, mediante un índice, los datos de las PDU que deben retransmitirse. Este índice debe de ser capaz de identificar plenamente cada una de las PDU que procesan los AcTMs. La forma de identificar plenamente las PDU es incluir en el índice el VPI/VCI de la conexión a que pertenece cada PDU. Además introducimos también el campo PDUid que, como hemos explicado en el apartado anterior, nos permite identificar cada una de las PDU de una conexión concreta. Ante la posibilidad de que en un mismo conmutador se

[1] ABR genera una célula RM cada 32 células de datos o cada 100 ms.

multiplexen células de diferentes conexiones que hayan decidido emplear el mismo valor de VPI/VCI, por provenir de diferentes usuarios emisores, hemos incluido en el índice el campo relativo al puerto de entrada (Port) por el que las fuentes acaban llegando a cada uno de los conmutadores. Así, el índice que incluimos en el campo reservado de las células RM esta formado por los siguientes valores: /Port/VPI/VCI/PDUid/ que constituyen una perfecta clave primaria, tanto en los accesos a la memoria DMTE, como en las tablas de E/S asociadas a cada uno de los puertos de salida de los conmutadores activos.

TABLA 9.1

CAMPOS Y SUS POSICIONES EN LAS CÉLULAS RM [2]

Campo	Octeto	Bit(s)	Descripción
Cabecera	1-5	Todos	RM-VPC:VCI=6 y PTI=110; RM-VCC:PTI=110
ID	6	Todos	Identificador de Protocolo
DIR	7	8	Dirección
BN	7	7	Célula BECN
CI	7	6	Indicación de Congestión
NI	7	5	No incremento
RA	7	4	Solicitud/Acknowledge
Reservado	7	3-1	Reservado
ER	8-9	Todos	Velocidad explícita de células
CCR	10-11	Todos	Velocidad actual de células
MCR	12-13	Todos	Mínima velocidad de célula
QL	14-17	Todos	Longitud de cola
SN	18-21	Todos	Números de secuencia
Reservado	**22-51**	**Todos**	**Identificador Port/VPI/VCI/PDUid**
Reservado	52	8-3	Reservado
CRC-10	52 53	2-1 Todos	CRC-10

Debemos destacar que la posibilidad de pérdida de las células BRM es la que causa que la fiabilidad no sea total. Es decir, las células BRM se emplean para recuperar pérdidas, sin embargo, nada nos garantiza que las propias RM puedan perderse por congestión o por errores de conmutación. Podría conseguirse la fiabilidad total aplicando la variante +ACK de ARQ en lugar de NACK, pero ya conocemos que de esta forma afectaríamos de forma muy importante al goodput de la red, haya o no congestiones en la misma. Nuestra intención es, por tanto, conseguir el mayor grado posible de GoS, pero sin renunciar al aprovechamiento de las prestaciones de la red.

9.5. IMPLOSIÓN EN LAS FUENTES DE TRÁFICO

Los protocolos de transporte de alta velocidad para aplicaciones multimedia mp imponen nuevos cambios incluyendo el soporte para aplicaciones de medio continuo y soporte escalable para un gran número de participantes. El control de errores en los protocolos multicast actuales no escala bien debido a la sincronización de los mensajes de realimentación, ya sean debidos a ACK o a NACK desde los receptores del tráfico. Estos dos requerimientos tienen un importante impacto sobre los mecanismos de control de errores, los cuales no soportan actualmente recuperación de retardo-limitado y no pueden controlar la implosión. La implosión[2] es causada en los receptores (*Figura 9.2*) por la sincronización de los mensajes de realimentación desde los receptores en las conexiones mp.

El fenómeno de la implosión es un importante aspecto en comunicaciones escalables multicast. Este problema adquiere importancia en el control de congestión mp o en el transporte fiable. Debido a que los recursos son compartidos en el árbol multicast, las pérdidas experimentadas por diversos receptores tienden a estar fuertemente correlacionadas, dirigidas a una implosión de mensajes de solicitud de retransmisión hacia el emisor del tráfico. Este fenómeno puede ser evitado retardando los NACK hacia el emisor.

En las conexiones mp, el control de la implosión puede ser realizado por los receptores o por el emisor [3]. En el primero de los casos los receptores colaboran para controlar la implosión y en la segunda es el emisor el responsable de controlar directamente la información de realimentación que proviene de los receptores. La implosión controlada por los emisores es apropiada para redes orientadas a la conexión donde los receptores son aislados unos de otros y no pueden colaborar (por ejemplo, como en una conexión bidireccional p-mp).

[2] **Implosión**: (De explosión, con cambio de prefijo) .f. Acción de romperse hacia dentro con estruendo de las paredes de una cavidad en cuyo interior existe una presión inferior a la que hay fuera. (Diccionario RAE, Ed. 1992).

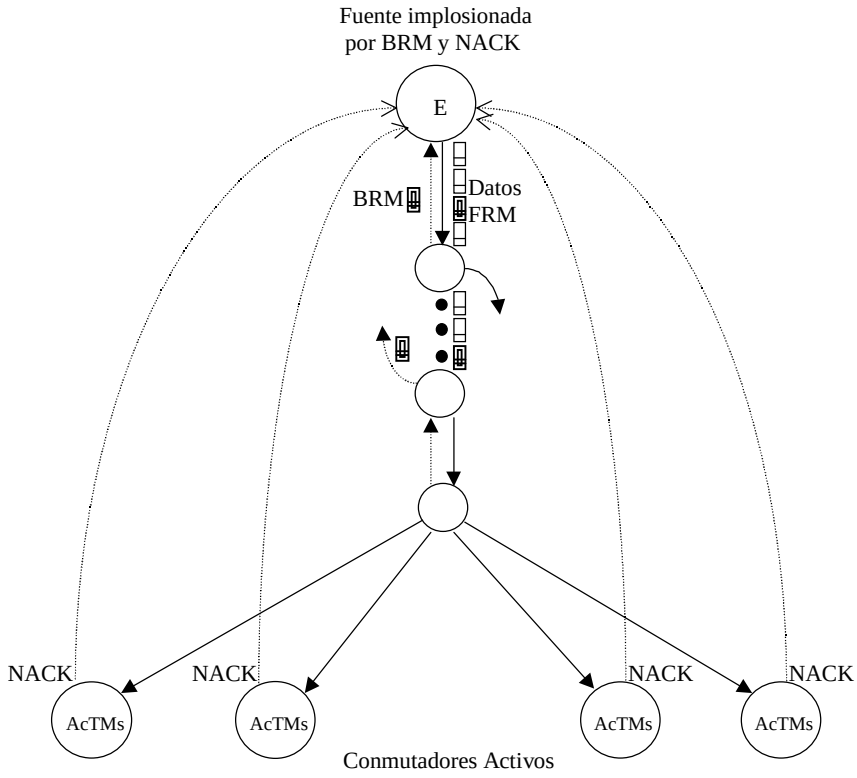

Figura 9.2. *Efecto de la implosión de células BRM y NACK sobre las fuentes*

Por otro lado, la implosión puede controlarse también estructurando los receptores en una jerarquía arborescente virtual, donde el emisor es la raíz, y donde los "hijos" sólo pueden enviar mensajes de realimentación a sus "padres". Esta estructura es también reconocida como adecuada para la consolidación de ACK que es requerida para el control de la implosión en multicast fiable. El trabajo [3] propone dos mecanismos para conseguir controlar la implosión basados, uno en los emisores, y el otro en los receptores. Esta referencia presenta también una interesante taxonomía sobre el control de la implosión, sin embargo, no se centra en ninguna tecnología concreta.

La referencia [4] presenta un algoritmos distribuido llamado DTRM (Deterministic Timeouts for Reliable Multicast) que permite computar de forma óptima los timeouts. Otro aspecto directamente relacionado con todo esto es el de la escalabilidad en comunicaciones mp donde todos los miembros del grupo son potencialmente emisores y receptores.

Precisamente, existe la creencia generalizada de que la naturaleza de la conmutación de células, en la que se basa ATM, impide la escalabilidad de las comunicaciones mp-mp que es importante para muchas aplicaciones.

La CoS ABR garantiza velocidad mínima, aporta justicia a las fuentes y minimiza la pérdida de células de datos, todo ello indicando periódicamente a las fuentes ABR la velocidad a la que pueden emitir. ABR usa un control de realimentación extremo-a-extremo en bucle cerrado. La realimentación se realiza desde los conmutadores hasta las fuentes a través de células BRM generadas periódicamente por las fuentes que son devueltas por los destinatarios. Las células RM que van en sentido del emisor a los destinos son conocidas como FRM (Forward RM), mientras las que viajan en sentido contrario de las fuentes a los emisores son llamadas BRM (Backward RM). Estas células BRM contienen la velocidad de células actual de las fuentes (CCR), además de otros campos que pueden ser usados por los conmutadores para ofrecer realimentación a las fuentes. La realimentación, por tanto, puede ser de uno o dos bits para indicar congestión, y/o la velocidad exacta a la que deberían transmitir las fuentes, llamadas ER (Explicit Rate). Cuando una fuente recibe una célula BRM, ésta computa su ACR (Allowed Cell Rate) usando su ACR actual, los bits de indicación de congestión (CI) y el campo ER (Explicit Rate) de la célula RM [2].

Las aplicaciones multimedia con requerimientos de retardo limitado son normalmente soportadas por las CoS CBR o rt-VBR. Sin embargo, el servicio ABR que fue originalmente diseñado para datos, puede soportar también aplicaciones multimedia bajo ciertas circunstancias.

Las aplicaciones multimedia tienen importantes requerimientos de ancho de banda y mientras unas (videoconferencia, VoD, colaboratorios, etc) necesitan garantías de retardo, otras (WWW, pizarras compartidas, B.D. gráficas, etc) pueden ser insensibles a los retardos pero no a las pérdidas. Para este tipo de aplicaciones se pensó en las CoS CBR y VBR, pero cuando los recursos de red son escasos, las conexiones con estas CoS suelen ser rechazadas, por lo que es preferible el servicio ABR. Incluso, aunque la conexión CBR o VBR sea aceptada, los usuarios pueden ser forzados a aceptar anchos de banda demasiado bajos [4], por lo que no pueden usar el ancho de banda aunque después esté disponible. Del mismo modo, si el ancho de banda comienza a ser escaso, la red podrá desconectar a las aplicaciones CBR y VBR por no poder mantener ni renegociar el contrato de tráfico inicial.

Sin embargo, la CoS puede garantizar el MCR que puede ser usado para aportar una aceptable QoS a las aplicaciones multimedia con múltiples destinatarios. Debido al bucle cerrado de realimentación de ABR, las colas de los conmutadores suelen ser pequeñas y la pérdida de células es baja. No obstante, aunque ABR fue pensado para aplicaciones de datos insensibles a los retardos, en el caso de usarse para aplicaciones multimedia, debe minimizarse la variación en los retardos y en la QoS. Pero uno de los aspectos más importantes del uso de ABR como servicio para las aplicaciones multimedia es que éstas suelen ser multicast (p-mp, mp-mp y pm-p), lo cual acaba afectando a la gestión de las colas de los conmutadores y, sobre todo, lleva aparejado el efecto de la implosión. Este efecto negativo [5] sobreviene cuando la información de realimentación que llega a los emisores incrementa en proporción al número de solicitudes (receptores). Por esto es necesario buscar soluciones a este problema para poder controlar la información que las fuentes de tráfico reciben desde las hojas de sus árboles de distribución (*Figura 9.2*).

Se han propuesto múltiples soluciones al problema de la implosión:

- Las células RM son difundidas a todas las ramas y enviadas hacia atrás en dirección al emisor, llevando la velocidad menor de todas las ramas. Este mecanismo aporta un cierto grado de justicia, pero tiene el problema de la falta de sincronía en la llegada de la información de realimentación y de cambios en las velocidades. No obstante, se han propuesto múltiples soluciones para evitar este problema.
- Una de esas soluciones es un esquema donde sólo se envían células BRM hacia el emisor si estas BRM han sido recibidas desde todas las ramas. Sin embargo, cuando la red está sobrecargada los conmutadores pasan rápidamente las BRM sin esperar la llegada de todas las ramas. De este modo se solventa el problema de consolidación citado antes, pero se mantiene el problema de implosión cuando la red está sobrecargada.

Por otro lado, la CoS permite minimizar los retardos, las pérdidas y las variaciones en la QoS. Durante la conexión, la longitud de las colas depende del valor inicial de ICR veces el RTT. Además, las fuentes pueden influirse en la determinación de la cola inicial. Durante los periodos *steady state*, debido al control del bucle de realimentación cerrado de la CoS ABR las colas de la red son pequeñas, por lo que suelen construirse sólo en los sistemas finales [5]. La pérdida de células es baja para ABR y puede usarse la codificación de AAL o las técnicas de control de errores FEC para hacer las aplicaciones mas resistentes a las pérdidas [5].

En cuanto a los retardos de encolado, éstos pueden ser controlados usando una función de control de encolado dinámico como la propuesta en el estándar por ERICA o ERICA+. ERICA+ utiliza una función de control de cola *f(q)* para calcular el ancho de banda que debe ser asignado como *f(q)* veces el ancho de banda total disponible. El valor de la *f(q)* depende de la longitud de la cola *(q)* de cada conmutador. Por tanto, usando una buena función de control de colas puede conseguirse un buen control de la longitud de las colas y por tanto un adecuado control del retardo en los periodos de *steady state*.

La literatura en torno a este problema es relativamente extensa, sin embargo, en el caso de ATM todas las propuestas se basan en los mecanismos descritos anteriormente, o en variantes de éstos. Como todos estos mecanismos se apoyan, o bien en la ruptura de sincronización en los mensajes de realimentación, o en la agrupación de los procedentes de ramas comunes del árbol de distribución, en realidad no acaban evitando de forma definitiva los NACK que se generan en el caso de congestiones, de pérdidas o de células erróneas, aunque alivian en gran medida los efectos de las BRM de ABR. En nuestro caso es importante destacar que nuestra propuesta se encarga de solventar la implosión debida a las congestiones, ya que los NACK son resueltos de forma local a los conmutadores congestionados en lugar de tener que ser solventados con retransmisiones desde los emisores.

La *Figura 9.3* presenta la aportación de nuestras investigaciones con respecto al problema implosivo que ilustraba la *Figura 9.2*. Suponemos que los cinco nodos activos extremos de la conexión son todos receptores de la misma fuente de tráfico E, aunque los dos nodos extremos de la derecha reciben el flujo desde el conmutador intermedio C4. Podemos observar cómo en la *Figura 9.3* los conmutadores AcTMs que bifurcan

varias ramas del árbol de distribución pueden atender las solicitudes de retransmisión sin permitir que la implosión acabe llegando al nodo emisor. Es decir, el primer intento por localizar una PDU perdida consiste en su búsqueda en el padre del conmutador congestionado recorriendo el árbol de distribución en profundidad y en sentido contrario al del tráfico. Sin embargo, cuando en profundidad no se localiza una PDU puede optarse por el recorrido en anchura del árbol de distribución, tal como puede observarse en la *Figura 9.3*. Es decir, si el conmutador C3 no contiene ya la PDU que se ha de retransmitir, ésta es solicitada al conmutador activo C4 (a través del C2 y recorriendo el árbol de distribución en anchura en lugar de en profundidad) para evitar que la implosión acabe llegando al nodo emisor. Podemos observar que, además de la implosión sobre la fuente de tráfico, también se evita la sobrecarga de la red con NACK.

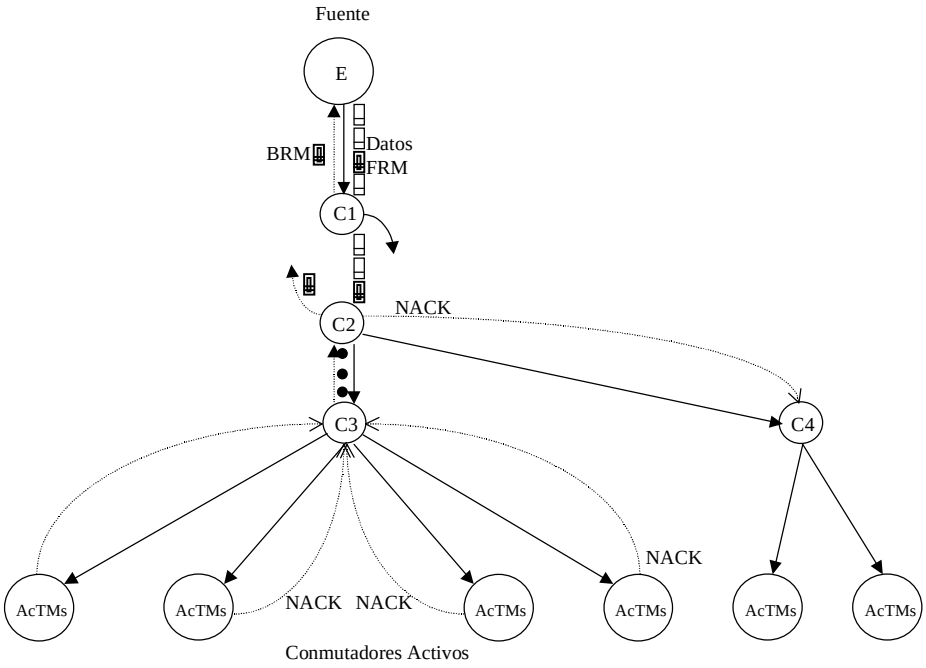

Figura 9.3. *Control de la Implosión de forma local*

La *Figura 9.3* muestra por tanto cómo la fuente de tráfico puede delegar la atención de las retransmisiones en los conmutadores activos más cercanos (C2 y C3) a los problemas de congestión que se encargan de evitar el problema de implosión.

Nuestra propuesta se basa en una solución jerárquica de la implosión pero, en lugar de ser resuelta por protocolos de la capa de transporte como en [3], en nuestro caso aprovechamos las características programables de la arquitectura TAP. Además, en nuestro caso se evita también la implosión en los conmutadores activos comparativamente a las propuestas de [3] donde se atienden retransmisiones generales de subárboles aunque sólo sea un receptor de los *N* que dependen de ese subárbol. Es decir, TAP atiende las peticiones individualizadas sin agrupar nodos pertenecientes al mismo segmento de red. Hemos de destacar que el problema de la implosión no es nuestro objetivo principal, sino que en realidad, este es uno de los problemas que quedan resueltos con la introducción de nuestra propuesta en las redes ATM. Las simulaciones realizadas presentan muy buenos resultados en cuando al porcentaje de PDU recuperadas cuando se provocan las congestiones, por lo que podemos garantizar que, además de la optimización en las recuperaciones locales evitando los retardos RTT entre receptores y emisores, el throughput no experimenta degradación, ni tampoco las fuentes de tráfico experimentarán implosión en el caso de conexiones multipunto. De este modo, conseguimos aportar una solución óptima a la implosión desde los siguientes puntos de vista:

- El retardo es menor en las retransmisiones solventadas jerárquicamente, que en las soluciones extremo-a-extremo. Esta mejora de la latencia ha sido estudiada en el capítulo anterior.
- La atención local no afecta al rendimiento de la red, ya que sólo se atienden las retransmisiones cuando existe suficiente tiempo de inactividad en los conmutadores activos.
- La solución local de las retransmisiones permite el aislamiento del resto de los enlaces de red no congestionados, lo que permite optimizar el throughput.

- Minimización del intercambio de mensajes en la red que contribuye a evitar la implosión, no sólo en los emisores del tráfico sino también a los conmutadores que intervienen en el VPI/VCI entre emisor y receptores.
- La escalabilidad del mecanismo usado depende de los recursos hardware que se introduzcan en los AcTMs. Las simulaciones demuestran que a mayor memoria DMTE en la arquitectura, mayor es la probabilidad de recuperación local de PDU congestionadas.
- Transparencia para los receptores del tráfico que no necesitan participar en el mecanismo de recuperación.

9.6. FRAGMENTACIÓN DE LAS PDU

Uno de los efectos colaterales provocados por la congestión de los conmutadores es, como hemos visto en el *Capítulo 4*, la fragmentación de las PDU que han experimentado congestión. Por tanto, además de la pérdida de células pertenecientes a una misma PDU (que acabarán dando como resultado una PDU corrompida en el destinatario cuando sea aplicado el CRC de AAL-5), se produce la sobrecarga innecesaria de la red con los fragmentos de esas PDU que podrían haberse descartado completas en cuanto se detecte la congestión. Como sabemos, este es uno de los objetivos de varios algoritmos entre los que destaca EPD [6].

Sabemos que los mecanismos de control de la fragmentación de paquetes se basan en aplicar una correcta política en los umbrales del buffer de los conmutadores. De este modo, ante la inminencia de una congestión lo que se hace es rechazar la entrada completa de una PDU ante el riesgo que ésta sea aceptada en parte y se envíen al siguiente conmutador algunas de sus células, pero otra parte de las mismas no tengan cabida en el buffer y sean descartadas generando así una PDU fragmentada. Podemos entender de forma clara que la fragmentación afecta muy negativamente al rendimiento de la red, que se ve obligada a transferir células innecesariamente cuando podrían ser descartadas, y por otro lado, se requerirá la retransmisión e-e de las PDU que han llegado fragmentadas. Se pierde por tanto ancho de banda de la red y también tiempo de proceso, ya que las pérdidas pueden ser solventadas antes de lo que se está haciendo actualmente. La *Figura 9.4* presenta el problema de la fragmentación de una PDU. Puede observarse cómo a la llegada de las PDU las número 1 y 2 caben perfectamente en el buffer, mientras de la PDU 3 se aceptan las dos primeras células, pero la célula 3 llega cuando el buffer está lleno por lo que será descartada. La fragmentación de la PDU 3 se observa a la salida donde puede comprobarse que las células 1 y 2 siguen adelante, mientras la célula 3 ya no aparece. De este modo la PDU llegará a los siguientes conmutadores hasta ser recibida por el destinatario del tráfico, que al aplicar el CRC detectará la PDU 3 como corrupta y deberá solicitar su retransmisión con un NACK desde el emisor.

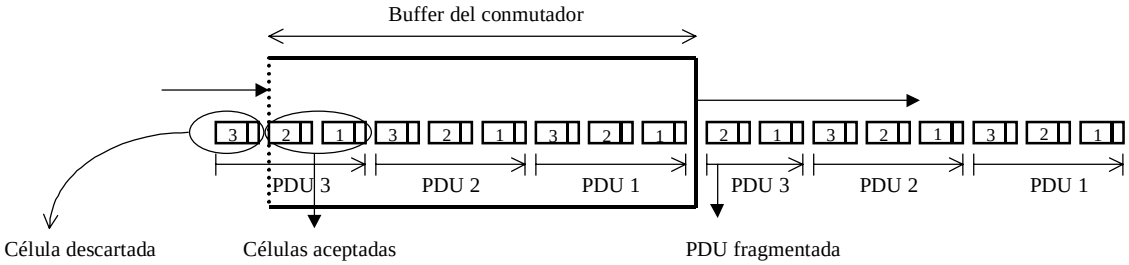

Figura 9.4. *Fragmentación de una PDU*

En la arquitectura TAP conseguimos evitar la fragmentación de las PDU mediante el establecimiento de un umbral dinámico que es controlado por el agente CCA sobre el que hemos implementado el algoritmo EPDR. Una vez establecido el valor del umbral, las PDU que no tengan cabida son completamente descartadas para evitar su fragmentación y, automáticamente, se pone en marcha el mecanismo de recuperación de esas PDU que son solicitadas mediante la generación de una célula BRM (de las descritas en el punto anterior) que se envía al conmutador previo. La *Figura 9.5* ilustra el funcionamiento de EPDR donde puede observarse una PDU que genera congestión y su solicitud de retransmisión. En este caso podemos ver el efecto del umbral (U) que es controlado por el agente CCA. Las PDU 1 y 2 tienen cabida en el buffer porque no alcanzan el valor de U; sin embargo, la última célula de la PDU 3 sobrepasa el valor del umbral, por lo que será descartada completa para evitar el problema de la fragmentación que hemos podido observar en la *Figura 9.4*. Así, sólo siguen adelante las PDU 1 y 2, mientras la número 3 será solicitada como retransmitida inmediatamente mediante una solicitud del agente RCA que será el encargado de generar la

célula RM con la identificación de la PDU descartada. Esta BRM es enviada en sentido contrario al flujo de datos al conmutador activo previo que se encargará de retransmitirla nuevamente si aún la conserva en su memoria DMTE y se dispone de tiempo de inactividad en la fuente de tráfico.

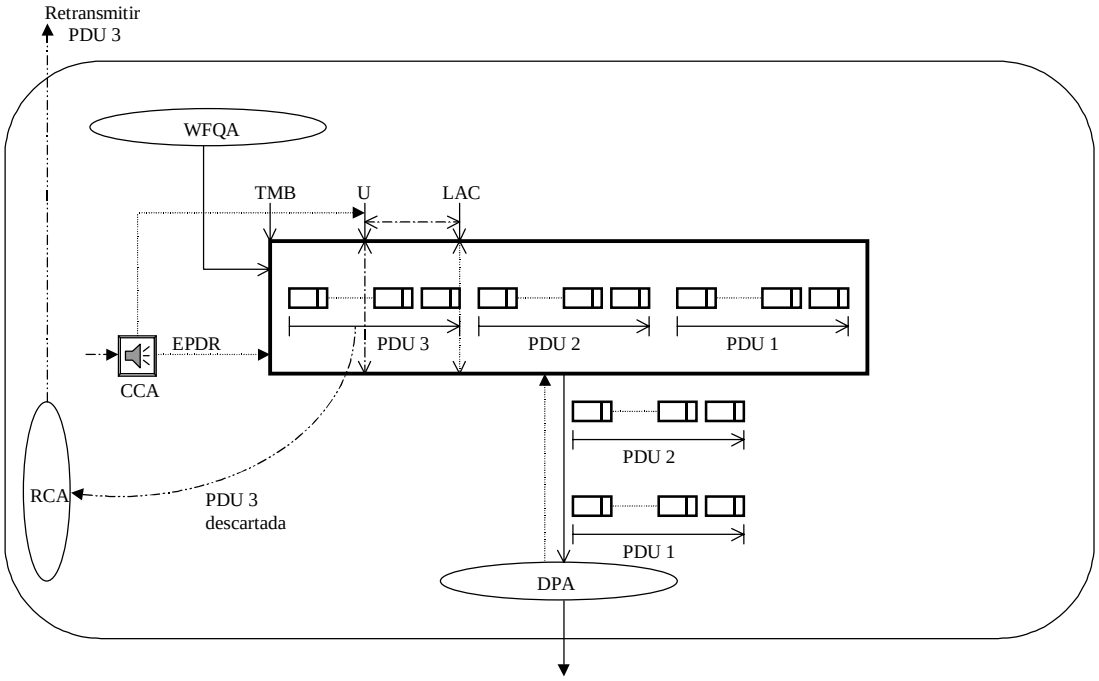

Figura 9.5. *PDU descartada y recuperada sin fragmentación*

9.7. INTERLEAVING DE CÉLULAS EN LOS PUERTOS DE SALIDA

Otro efecto pernicioso que se produce en las redes ATM actuales es el del *interleaving* que ya fue presentado en el *Capítulo 5*. La mezcla de las células pertenecientes a diversas conexiones en los puertos de salida de los conmutador afecta directamente al buen comportamiento de la red. En nuestro caso, este problema acaba también afectando a la GoS que queremos ofrecer a las conexiones garantizadas. Es decir, si una de nuestras aspiraciones es la de conseguir que determinadas conexiones puedan disponer de garantía en sus transferencias, y estas conexiones privilegiadas se ven obligadas a compartir los puertos de salida de los enlaces comunes con otras conexiones sin requerimientos de GoS, esto no puede más que afectar negativamente a estos objetivos.

Cuando las células de las conexiones llegan a la matriz de conmutación de un conmutador ATM, éstas son multiplexadas a sus correspondientes puertos de salida, y si no aplicamos una política de atención cuidadosa en las fuentes, acabarán mezclándose a la salida las células del tráfico con requerimientos de GoS con las células que no necesitan esos requerimientos.

Sabemos ya que existen diversas técnicas para evitar esta mezcla, y hemos destacado que el VC Merge es la más adecuada de las propuestas existentes en la actualidad [7]. VC Merge se caracteriza por evitar el *interleaving* aplicando un mecanismo de asignación de valores de VCI iguales a los flujos que van a ser multiplexados por un enlace común. La *Figura 9.6* ilustra el funcionamiento de las tablas de conmutación sin aplicar VC Merge, y el efecto producido al aplicarlo. Como podemos observar, si no se aplica ninguna política en la tabla de VCI, a la salida del conmutador se va produciendo la mezcla continuada de células pertenecientes a distintos flujos. Sin embargo, al aplicar VC Merge se asigna un mismo valor de VCI de salida a todas las conexiones que compartirán un mismo enlace, de forma que se envían a la salida del conmutador las células de cada conexión de forma separada a las de otras conexiones.

En la arquitectura TAP la aplicación de una cuidada política de atención y separación de los flujos acaba aportando importantes ventajas, ya que, aunque la unidad de conmutación es siempre la célula, el concepto PDU como agrupación de un conjunto de células pertenecientes a una misma conexión, es la base de nuestra filosofía de recuperación de pérdidas en los conmutadores congestionados. En realidad, aunque nuestra forma de evitar la mezcla de flujos se inspira en VC Merging, nuestro planteamiento tiene importantes diferencias con la técnica representada en la *Figura 9.6*.

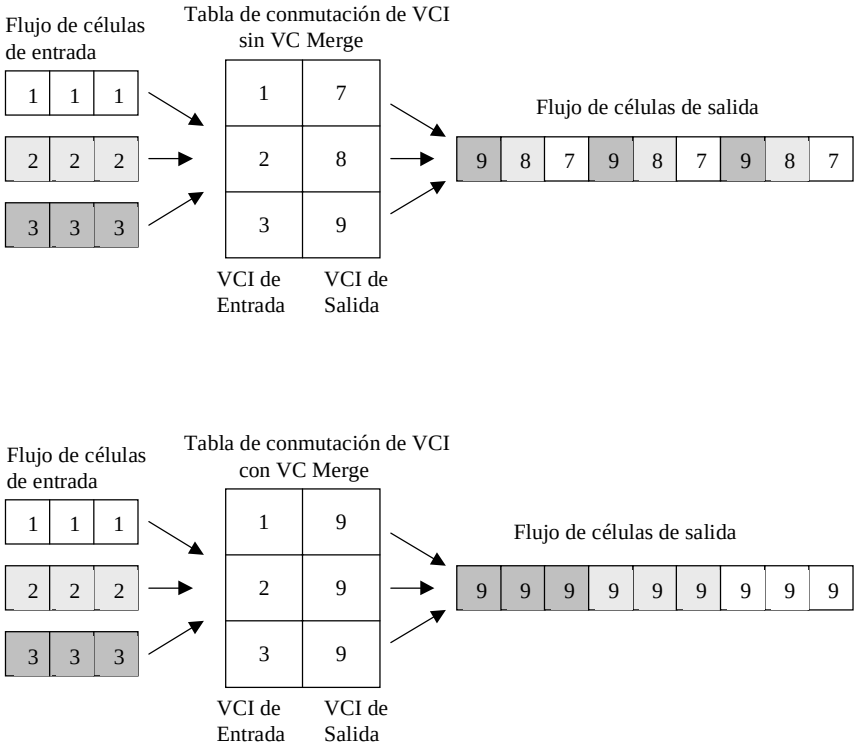

Figura 9.6. *Efecto de la aplicación de VC Merge*

Como hemos podido comprobar en el apartado anterior, EPDR nos permite eliminar la fragmentación de las PDU que se almacenan en el buffer, por lo que podemos garantizar que las PDU son unidades de información continua que van a ser enviadas completas a su correspondiente puerto de salida, con lo cual garantizamos que no aparece el *interleaving* con células pertenecientes a otras conexiones. Es decir, una vez que una PDU es sacada del buffer, es tratada completa desde su inicio hasta llegar al final de la misma, con lo cual se garantiza su orden de atención. Esta forma de actuar puede observarse en la *Figura 9.7*, donde podemos comprobar cómo el tráfico de cada conexión (VPI/VCI) llega al buffer del conmutador AcTMs a través de las colas de entrada en forma de PDU. Las PDU que han sido aceptadas completas en el buffer (evitando su fragmentación) son también procesadas completas a su correspondiente puerto de salida (evitándose el *interleaving* con otras conexiones).

Para poder realizar esta labor contamos con la posibilidad de usar desde el protocolo TAP una operación atómica que nos permite tomar las PDU completas del buffer del conmutador y poner todas sus células en su correspondiente puerto de salida. De este modo, en una misma operación atómica sólo pueden tratarse células pertenecientes a una sola conexión garantizando que todas las células de una PDU son enviadas como un flujo contínuo a la salida sin oportunidad para provocar la mezcla con otras células de conexiones, sean privilegiadas o no. Esta operación atómica depende del agente DPA que es el encargado de solicitar al buffer el envío de las PDU o de las células independientes.

9.8. EXIGENCIAS DE LA TECNOLOGÍA

En ocasiones se ha argumentado que la tecnología ATM parte de dos premisas básicas como son obtener la máxima velocidad de conmutación posible y con una mínima complejidad en los conmutadores. No obstante, hemos de destacar que la literatura está repleta de propuestas que se esmeran en aplicar técnicas y métodos capaces de ofrecer a las aplicaciones las mejores posibilidades en cada caso. Del mismo modo, la propia tecnología va evolucionando e incorporando muchas de estas propuestas que, en algunos casos, se distancian levemente de las dos citadas premisas. No debemos perder de vista que ATM se caracteriza, entre otras muchas ventajas, por su posibilidad de tratamiento del tráfico en función de la clase de servicio que demanden las aplicaciones que se usan. Por ello existen múltiples propuestas pensadas para caracterizar el tráfico y, en función de esto, aplicar unas u otras calidades de servicio. En muchos casos, por tanto, puede estar justificado aplicar mecanismos que añadan una relativa complejidad a los conmutadores, así como un relativo coste computacional.

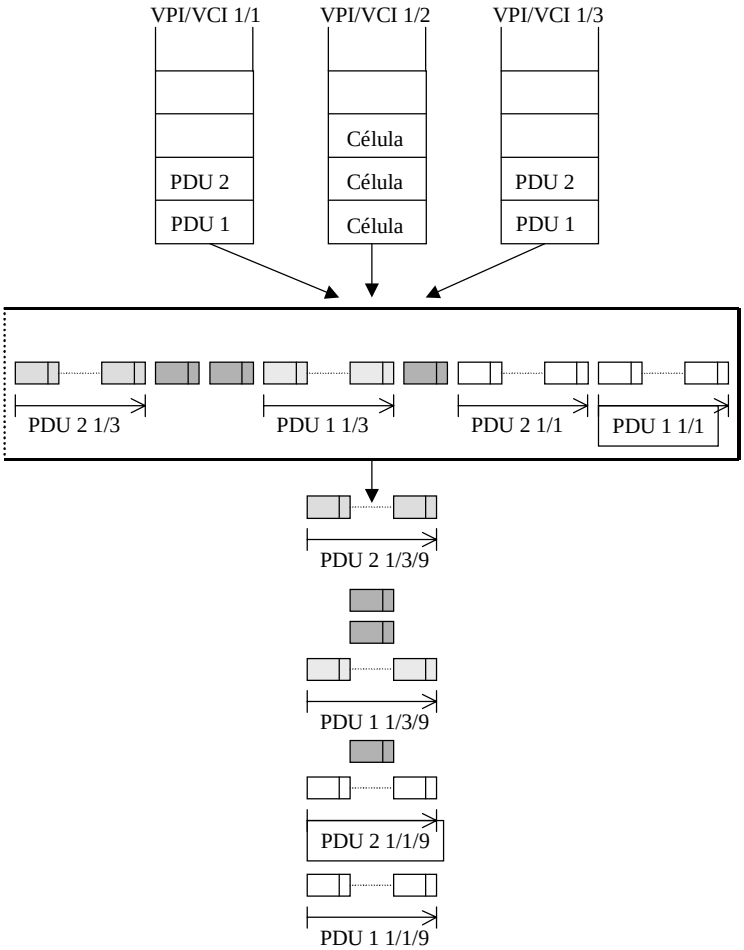

Figura 9.7. *Efecto del VC Merge sobre las PDU*

Uno de estos casos es precisamente el tipo de aplicaciones para el que va destinada nuestra propuesta. La GoS tal como la hemos definido está pensada para aplicaciones, principalmente de datos, que tienen unos especiales requerimientos de transferencias garantizadas, sin por ello afectar al rendimiento del resto de conexiones que están usando la red. El tráfico UBR y ABR es, por tanto, para el que está indicado la arquitectura TAP, ya que sabemos que este tipo de tráfico de datos tiene unos especiales requerimientos en cuanto a la fiabilidad sin por ello renunciar al throughput ni tampoco a los retardos. No debemos olvidar que en muchos casos, las aplicaciones no sólo esperan recibir de la red unos elevados anchos de banda, sino que también es importante para ellas el tener garantizado un mínimo caudal sostenido que no se vea afectado por las pérdidas que la propia tecnología asume como algo factible.

No obstante, hemos de destacar que aunque puede parecer que la complejidad de TAP es excesiva para resolver todas las limitaciones que acabamos de exponer en este capítulo, en capítulos siguientes aclararemos que el coste computacional para implementar las soluciones propuestas es perfectamente asumible en nuestra arquitectura y protocolo. Por ejemplo, el hecho de usar los números de secuencia en las PDU no añade ninguna complejidad a los actuales mecanismos de generación de estas unidades de transferencia. En nuestro caso únicamente nos encargamos de mantener la secuencia y aprovechar dos campos de escasa utilidad en los estándares. Para no afectar a los recálculos de CRC, los números de secuencia de las PDU son válidos e-e, con lo cual evitamos también la carga innecesaria de la red.

Por otro lado, el mecanismo de generación de las BRM en el caso de congestiones tampoco añade ninguna complejidad adicional a las propuestas existentes. De hecho, las células RM son, como ya sabemos, una propuesta de la CoS ABR que hemos decidido incluir en nuestra arquitectura, sólo que en nuestro caso no usamos una frecuencia fija de BRM, como hace ABR, y de este modo sólo se consume coste computacional y recursos de red cuando aparecen las congestiones. Esto es en sí una ventaja añadida sobre la propia CoS ABR. De echo, la introducción de las BRM lo que consigue es optimizar el goodput de la red cuando experimenta pérdidas. Como ya se ha demostrado en el *Capítulo 8*, la aparición de congestiones afecta negativamente a la red que se ve obligada a solventarlas mediante las retransmisiones e-e que TAP

evita mediante retransmisiones locales p-p gracias al uso de las BRM.

El uso de las células BRM también está justificado por el beneficio que aportan para evitar el efecto negativo de la implosión. En las conexiones p-mp está también justificado el uso de TAP ya que el funcionamiento de la red puede verse degradado de una forma muy importante cuando aparecen las congestiones. Este es, por tanto, otro de los argumentos que puede justificar la relativa complejidad añadida en la labor de conmutación de TAP.

Por otro lado, la justificación de las técnicas que evitan la fragmentación del tráfico ha sido ya expuesta en capítulos precedentes. Conseguir evitar este problema ya justifica por sí mismo la introducción de una política de gestión del buffer. En nuestro caso sólo empleamos una variable para fijar el valor óptimo del umbral que permita a EPDR descartar el menor número posible de PDU, pero reduciendo al mínimo el número de PDU fragmentadas. El único coste adicional a EPD es la generación de la célula BRM cada vez que se produce una congestión. Consideramos que este relativo coste computacional y de complejidad añade más que importantes ventajas, ya que permite recuperar un gran número de PDU que de otro modo se pierden y requieren retransmisiones e-e como ya sabemos.

Nos enfrentamos también al *interleaving* de las células con un sencillo mecanismo basado en VC merge, pero mucho más simplificado que las propuestas descritas en la literatura. En nuestro caso proponemos la utilización de operaciones atómicas que permitan atender las PDU de forma íntegra e individualizada, lo que en nuestro caso no añade mayor complejidad como en otras propuestas, ya que TAP mantiene en el buffer del conmutador las PDU perfectamente estructuradas por lo que no es necesario su generación para conseguir el VC merge. De este modo solo requerimos de una sencilla operación atómica que se encargue de cambiar los valores de punteros de memoria para dirigir las PDU (previa segmentación) a sus correspondientes puertos de salida. Es también este un coste asumible por las ventajas que acaba aportando al tráfico.

Otro aspecto destacable es el hecho de que las retransmisiones locales sólo son realizadas en el caso que las fuentes de tráfico tengan suficiente tiempo de inactividad para atenderlas. De este modo garantizamos que las retransmisiones nunca afecten negativamente al tráfico normal de datos. Antes de solicitar una retransmisión el agente RCA consulta el tiempo agregado que tiene la fuente afectada para determinar si va a ser factible atender una retransmisión. Así obtenemos dos ventajas, por una lado no se afecta negativamente al tráfico normal de datos, y por la otra eliminamos las solicitudes de retransmisión cuando la probabilidad de éxito en la recuperación de la DMTE de conmutadores previos es muy baja. Es decir, cuando la fuente es muy activa, y tiene pocos estadios de inactividad con respecto al ancho de banda del enlace que usa, existe una probabilidad muy baja de localizar las PDU perdidas en la memoria dinámica de conmutadores anteriores, por el propio dinamismo del tráfico y la memoria. En estas situaciones no se solicita la retransmisión para evitar sobrecargar la red innecesariamente. Estos extremos tendremos ocasión de justificarlos con fuentes ON/OFF en capítulos siguientes.

En lo relativo a los componentes hardware necesarios en los conmutadores activos hemos de destacar también que las necesidades no van más allá de un aceptable tamaño de memoria para la DMTE, así como para las tablas de entrada salida asociadas a los puertos de cada conmutador. En el *Capítulo 10* tendremos también ocasión para justificar y explicar todos estos componentes específicos de nuestra arquitectura.

9.9. CONCLUSIONES

En este capítulo hemos tenido ocasión de agrupar los problemas a los que se enfrenta la actual tecnología para ser capaz de ofrecer la GoS que proponemos y que hemos definido como derivada de los conocidos parámetros generales de QoS. Partimos de la idea que no buscamos fiabilidad en las conexiones, sino que nuestra aspiración es garantizar las transferencias de datos cuando aparecen congestiones en los conmutadores. Por esto delegamos la fiabilidad en los actuales mecanismos de ATM que emplea el campo HEC para garantizar las cabeceras de las células, y usa un CRC-32 para ofrecer fiabilidad a las PDU de AAL-5. Sin embargo, otros problemas requieren de propuestas adicionales para conseguir nuestros objetivos. De este modo hemos justificado el uso de los números de secuencia en las PDU, así como la generación de células BRM para atender las retransmisiones. La implosión y la fragmentación son dos indeseables fenómenos que tienen su origen en la aparición de congestiones, por lo que hemos propuestos sendos mecanismos para aliviar su efecto. También el *interleaving* acaba afectando negativamente el throughput, por lo que hemos decidido evitarlo aprovechando el hecho que la propia filosofía de funcionamiento de TAP no implica mayor complejidad para hacerlo. La complejidad y el coste es otro de los argumentos que debemos de manejar para justificar el uso de TAP, y como tal hemos argumentado que es asumible y justificado el coste introducido por la implementación de TAP. En siguientes capítulos demostraremos con más detalle algunos de los aspectos relativos al coste computacional de nuestras propuestas.

REFERENCIAS

[1] Recommendation I.363.5, "B-ISDN ATM Adaptation Layer Specification, Type 5," *ITU-T*, 08/1996.

[2] The ATM Forum Technical Committee. Traffic Management Specification Version 4.0 *af-tm-0056.000*, April 1996.

[3] Papadopoulos, Christos; Parulkar, Guru M.; "Implosion Control for Multipoint Applications," *Proceedings of the 10th Annual IEEE Workshop on Computer Communications*, September 1995.

[4] Matt Grossglauser Two Issues in Multicast: Feedback Implosion Avoidance and ATM Group Communication called SEAM (Scalable and Efficient ATM Multicast).

[5] Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal, and Mukul Goyal, "QoS and Multipoint Support for Multimedia Applications over the ATM ABR Service," *IEEE Communications Magazine*, (1999).

[6] Allyn Romanov and R. Oskouy, "A performance enhancement for packetized ABR and VBR+data," *AF-TM 940295*, (1994).

[7] I. Widjaja, and A. I. Elwalid, "Performance issues in VC.merge capable switches for multiprotocol label switching," *IEEE Journal on Selected Areas in Communications*, pp.1178-1189, (1999).

CAPÍTULO 10

DESCRIPCIÓN DETALLADA DE LA ARQUITECTURA DISTRIBUIDA Y MULTIAGENTE

10.1. VISIÓN GENERAL

En la *Parte I* de esta tesis hemos tenido la ocasión de revisar los trabajos relacionados con nuestras investigaciones donde, además, hemos adelantado los aspectos más importantes de nuestras aportaciones en cada uno de los capítulos correspondientes. En la *Parte II* hemos presentado nuestras motivaciones generales y también hemos identificado las limitaciones de la tecnología ATM con las que nos enfrentamos para conseguir la GoS donde, también, hemos destacado las soluciones que proponemos para conseguir nuestros objetivos. En este capítulo realizaremos la descripción detallada de la arquitectura TAP para explicar sus componentes principales así como su funcionalidad y también el funcionamiento de los conmutadores activos AcTMs que la soportan.

Así, en este primer apartado del *Capítulo 10* retomamos la introducción realizada en el *Capítulo 6* para obtener una visión general sobre la arquitectura de los AcTMs que será sucesivamente ampliada en los apartados siguientes del capítulo, donde serán justificados cada uno de los componentes, desde el punto de vista de las ventajas que aportan, así como desde las implicaciones que supone su inclusión en los conmutadores. Recordamos que las dos tareas principales de un conmutador ATM, o de un nodo *cross-connect*, son la traslación de los valores VPI/VCI de entrada a los correspondientes valores VPI/VCI de salida y el transporte de las células desde sus puertos de entrada hasta sus respectivos puertos de salida.

Nuestra propuesta de arquitectura se basa plenamente en el modelo arquitectónico propuesto [1] para la tecnología ATM, de forma que nuestro principal interés es soportar el tráfico ATM nativo y también el procedente de otros protocolos tan extendidos como TCP. Por esto, la arquitectura TAP aprovecha las posibilidades que le aportan los propios estándares para aportar soluciones que esos estándares no ofrecen a la propia tecnología. La *Figura 10.1* intenta aportar la visión más general de TAP aprovechando el propio modelo ATM, de forma que se vea de forma clara la situación de cada uno de los bloques que componen la arquitectura. Así, podemos comprobar cómo hemos sustituido la capa AAL-5 por nuestra EAAL-5 para aprovechar las ventajas que ésta nos aporta como ya hemos discutido. Las capas ATM y Física permanecen intactas; sin embargo, sobre la capa EAAL-5 se sitúa el protocolo TAP que está formado por un conjunto de algoritmos que permiten obtener nuestros objetivos. Por encima de TAP podemos encontrar alguno de los protocolos clásicos de capas superiores, aunque éstos no serían necesarios en el caso de emplear tráfico nativo ATM. La *Figura 10.1* nos permite observar también la posición que ocupa el subsistema SMA-TAP que abarca desde el Plano de Gestión hasta el Plano de Control del modelo tridimensional de ATM.

La arquitectura está formada por los aspectos que destaca la *Figura 10.1* y, además, debe estar soportada sobre los conmutadores activos AcTMs que deben estar específicamente equipados con los bloques hardware y código software que vamos a describir en los siguientes apartados. En líneas generales el software constituye el protocolo TAP en su conjunto, que es la unión del código que implementa cada uno de los agentes software y que controlan cada uno de los bloques hardware de los conmutadores que soportan el protocolo y, por tanto, la arquitectura completa. Sabemos también ya que los nodos extremos de la comunicación deben soportar completamente la arquitectura, mientras los conmutadores activos únicamente necesitan disponer de capa Física, ATM y parte de EAAL-5 para soportar la identificación de las PDU.

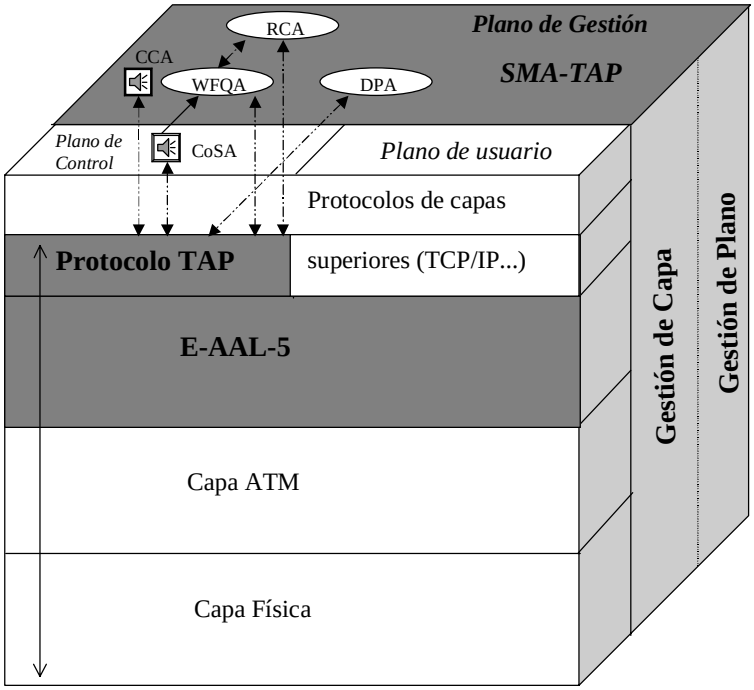

Figura 10.1. *Arquitectura TAP sobre el modelo arquitectónico estándar ATM*

Por otro lado, sabemos también que proponemos una VPN, por lo que no es obligatorio que todos los nodos que componen la red privada soporten la arquitectura TAP, para que se pueda disponer de GoS en las conexiones. Por tanto, los nodos que no implementen TAP podrán tratar el tráfico de la forma tradicional, sin la posibilidad de recuperar congestiones de forma local, pero sí de transferir las células RM estándares entre nodos AcTMs, para lo cual sólo necesitaremos adaptar levemente las labores de la señalización del estándar.

Tenemos ya una noción de los problemas que deseamos resolver con nuestra propuesta, por lo que debemos destacar que hemos incluido componentes específicos en la arquitectura para enfrentarnos a esos problemas. En líneas generales, podemos identificar (en la *Figura 10.2*) los siguientes bloques en la arquitectura de los nodos activos de la red, cada uno de los cuales intenta solventar problemas concretos de los identificados en el *Capítulo 9*:

- Colas de entrada, equipadas con un mecanismo justo de control de congestión que permita caracterizar el tráfico de entrada a los conmutadores mediante asignación de pesos.
- Buffer del conmutador, sobre el que se multiplexa el tráfico de las conexiones provenientes de las colas de entrada. Este buffer está gestionado con mecanismos de control de congestión que evitan la fragmentación de las PDU. El adecuado tratamiento del buffer nos permite también evitar el *interleaving*.
- Memoria dinámica DMTE, que realiza las veces de buffer temporal en el que cada AcTMs almacena el tráfico con requerimientos de GoS y desde donde se realizan las retransmisiones en el caso de aparecer congestiones.
- Tablas de Entrada/Salida, que están asociadas a cada uno de los puertos de los conmutadores AcTMs con dos funciones concretas. Por un lado las Tablas de E/S se encargan de mantener la relación entre los datos de entrada y de salida de cada una de las conexiones para evitar la posibilidad de valores repetidos en los VPI/VCI en fuentes diferentes. Por otro lado, estas tablas nos sirven como índice de acceso a la memoria DMTE en las operaciones de retransmisión locales.
- Los cuatro componentes anteriores son controlados por el subsistema SMA-TAP que ya hemos comentado que actúa como un sistema multiagente en el que, mediante un grupo de cinco agentes software, se realiza la supervisión y control de toda la actividad del conmutador. En realidad, el SMA-TAP representa el componente software que hemos desarrollado para el control de la equipación hardware que se ha indicado en los cuatro puntos anteriores. La mayor parte de las ventajas de este subsistema ya han sido explicadas en el *Capítulo 6*, aunque tendremos ocasión en este capítulo de explicar la función de cada uno de los agentes y las ventajas de su inclusión para disponer de conmutadores programables con funciones activas en la VPN que proponemos.

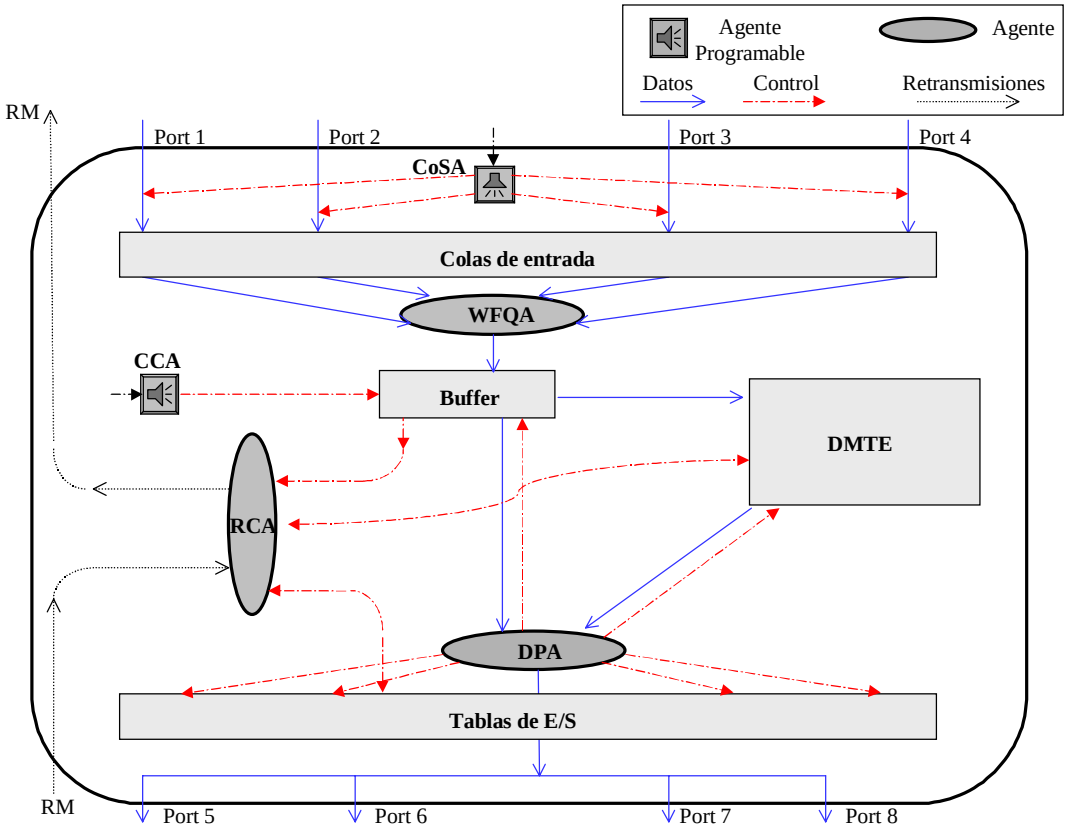

Figura 10.2. *Esquema de bloques de la arquitectura de los conmutadores AcTMs*

La *Figura 10.2* presenta una visión general de los conmutadores AcTMs con todos los componentes hardware anteriores así como cada uno de los agentes software que intervienen en la gestión del tráfico y control sobre cada uno de los componentes hardware. Podemos observar que el agente programable CoSA actúa sobre el tráfico de entrada que llega a las colas de entrada. Del mismo modo, el agente WFQA decide la política de atención de las colas de entrada en función de los pesos del tráfico para llegar al buffer del conmutador. El tráfico que llega al buffer es controlado según las características de los agentes CCA y DPA. Así el tráfico va dirigido a la memoria DMTE y a los puertos de salida. El agente DPA también controla los índices de las Tablas de E/S a medida que las PDU son enviadas a los puertos de salida de los conmutadores activos. Por último, el agente RCA genera y atiende las peticiones de retransmisión en caso de congestiones, teniendo relación también con las Tablas de E/S.

En las secciones que siguen describiremos cada uno de los módulos de la arquitectura del conmutador activo, destacando la relación entre cada uno de ellos, para pasar después a estudiar los flujos de datos y control dentro y fuera del conmutador.

10.2. CONMUTADORES ACTMS

Existen en la literatura múltiples arquitecturas de conmutadores ATM, aunque la mayor parte de ellas se basan en el uso de una matriz de conmutación donde confluyen las células de las conexiones de entrada que son conmutadas (tras asignarles sus correspondientes valores de VPI/VCI) a los puertos de salida. Sobre esta propuesta base de arquitectura se han realizado múltiples variaciones entre las que podemos situar TAP que, con la intención de solventar las problemáticas que ya conocemos, ha sido equipada con los módulos citados en el apartado anterior, los cuales van a ser seguidamente comentados en detalle.

10.2.1. COLAS DE ENTRADA

Los conmutadores que constituyen las redes ATM deben encargarse de mezclar los flujos de tráfico que provienen de fuentes diferentes y enrutarlos después hacia destinos distintos a través de *paths* de conmutación y enlaces de transmisión que las células de las diversas fuentes pueden compartir durante gran parte de su tránsito. Para que todo este proceso sea posible, los nodos de la red deben disponer de un almacenamiento temporal de las células en buffers de tamaño finito, de forma que el flujo de los datos en

cada momento pueda hacer que el tamaño de estas colas de almacén temporal puedan crecer y disminuir en su tamaño de forma dinámica. En realidad, este no es ningún mecanismo nuevo si lo comparamos con las redes de conmutación de paquetes clásicas, aunque lo realmente novedoso en el caso de ATM es la muy superior velocidad de conmutación (superiores a 622 Mbps en ATM frente a poco más de 256 Kbps en X.25). Estos requerimientos de elevada potencia de conmutación obliga a que cualquier propuesta que pueda hacerse para la labor de encolado del tráfico de entrada a los conmutadores deba ser optimizada en cuanto a los tiempos de acceso y de procesamiento.

Por tanto, las labores principales de los conmutadores de la red son las de ofrecer un almacén temporal de las células en tránsito hasta su destino (o al menos hasta el siguiente conmutador), reenrutar estas células desde este almacén al correspondiente puerto de salida del conmutador y, sobre todo, conseguirlo de la forma más eficiente y rápida posible, aunque en nuestro caso deseamos añadir una nueva prestación como la GoS para aportar esa fiabilidad que la red no aporta, pagando por ello un precio importante en prestaciones cuando el almacén temporal de las células experimenta las impredecibles congestiones. Veremos después que las técnicas de encolado en los nodos de la red juegan un papel fundamental en el diseño, operación y rendimiento de las redes ATM, por lo que existen una gran variedad de formas de solventarlo en los conmutadores, aunque los esquemas principales se dividen en las tres siguientes categorías (representadas en la *Figura 10.3*), que colocan las colas en los siguientes puntos de los conmutadores:

- En la entrada del conmutador, que se corresponde con la opción a) de la *Figura 10.3*, donde los buffers de células se sitúan a la entrada del conmutador. Si se emplean buffers FIFO, se producen colisiones cuando dos o más cabeceras de colas compiten simultáneamente por la misma salida, lo que provoca que una de las células pueda quedar bloqueada. Pero también quedan bloqueadas las células que siguen a la célula bloqueada, aunque no tengan la misma salida. Este tipo de desventaja puede solventarse con memorias RAM, aunque se requiere para ello un control de las colas más complejo. Sucesivamente se han ido proponiendo mejoras para este tipo de colas de entrada, y un ejemplo de ello es la propuesta que comentaremos en TAP.

- En la salida del conmutador, que se corresponde con la opción b) de la *Figura 10.3*, donde podemos observar que, en este caso, las colas de buffers se sitúan en los puertos de salida del conmutador. En este tipo, el elemento de conmutación consta de una matriz con buffers de salida y, sólo cuando la matriz opera a la misma velocidad que las líneas entrantes, se provocarán las colisiones; es decir, varias células compiten simultáneamente por el mismo puerto de salida. Este problema puede atacarse reduciendo el tiempo de acceso al buffer, o bajando la velocidad de la matriz de conmutación. En muchos casos el bloqueo interno se debe resolver con buffers adicionales.

- En los puntos de cruce (*crosspoints*) de la matriz de conmutación de los conmutadores, como puede observarse en el caso c) de la *Figura 10.3*. A este tipo de elementos de conmutación suele denominarse *butterfly* o de encolado central [2], y se caracterizan porque los buffers sólo se colocan en los puntos de cruce de la matriz, con lo que se previene que las células que compiten por puertos diferentes no se afecten mutuamente. Además, mediante un control lógico, puede elegirse qué buffer es el primero en el caso de que varias células o paquetes compitan por el mismo puerto de salida.

La situación c) de los buffer en los *crosspoint* supone disponer de una cola en cada uno de los puntos de cruce de la matriz de conmutación, lo que implica que hay que dividir el espacio de buffer total en N^2 colas diferentes, cuando podrían ser N como ocurre en las otras dos opciones citadas. Esta división de recursos de almacenamiento puede acabar provocando el incremento de la probabilidad de pérdida de células al incrementar el número de colas potencialmente congestionables, además de aumentar considerablemente la labor de gestión de las colas.

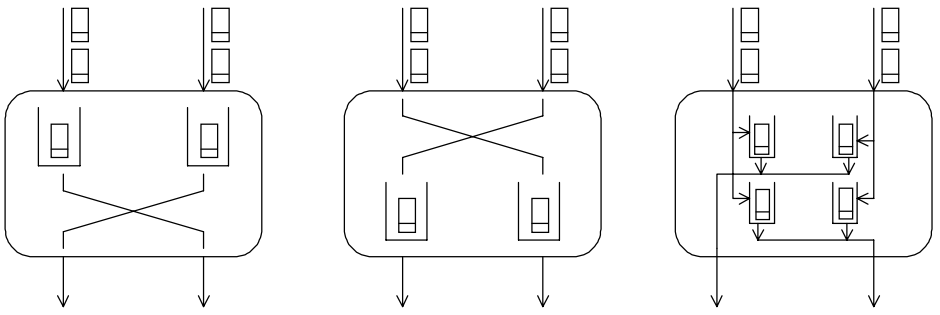

Figura 10.3. *Alternativas para situar las colas en los conmutadores*

La mayor parte de los estudios de estos tres diseños principales se inclinan por el uso de colas de entrada, aunque otros lo hacen por los de colas de salida si se combinan con la posibilidad de compartir el espacio de almacenamiento disponible entre todas las salidas [3]. Como sabemos ya, en nuestro caso hemos optado por el uso de colas de entrada para poder realizar un óptimo tratamiento de los flujos a la entrada de los conmutadores. Para nuestros intereses la gestión del tráfico a la entrada nos aporta importantes ventajas que han sido descritas a lo largo de toda esta tesis y, sobre todo, en el *Capítulo 4*. Si deseamos poder caracterizar el tráfico con pesos que además nos aporten la posibilidad de aplicar mecanismos justos de servicio de las colas, la mejor posibilidad es la de aplicar el buffering del tráfico en los propios puertos de entrada y antes de ser pasados por la matriz de conmutación. Además, la mayor parte de limitaciones de la tecnología y de las soluciones propuestas identificadas en el *Capítulo 9* apuntan a la posibilidad de poder tratar el tráfico en los puntos de llegada a los conmutadores. Debemos destacar que TAP aporta una novedad con respecto a las tres opciones representadas en la *Figura 10.3*, de forma que introducimos en la arquitectura de nuestros conmutadores un buffer (que puede ser identificado en la *Figura 10.2*) que es en realidad el punto de multiplexación del tráfico que llega desde las colas de entrada y que es donde solventamos las congestiones.

Una aspiración clave de TAP es la de aportar soluciones óptimas cuando aparecen congestiones en los conmutadores. Pues bien, con la intención de evitar la aparición de esas congestiones se propuso la utilización de colas de entrada que realizan las labores de buffer de entrada para el almacenamiento temporal de las células de entrada mientras son procesadas por la matriz de conmutación. La mayor parte de propuestas basan su trabajo en mecanismos de separación de flujos en las colas de entrada. Es decir, cada una de las conexiones dispone de una cola de entrada en la que se procesan separadamente células con valor de VPI/VCI diferentes, en un intento por realizar un tratamiento individualizado de cada una de las conexiones.

Como también hemos visto en capítulos anteriores resulta muy interesante la aplicación de técnicas que apliquen justicia en el procesamiento de las colas. El objetivo de estas técnicas es conseguir evitar las situaciones de inanición de las fuentes con menos requerimientos con respecto a aquellas que sean más ambiciosas en cuanto a los recursos de red que necesitan. Las técnicas de justicia son muy apropiadas en cualquiera de las situaciones, por lo que hemos decidido soportar esta posibilidad sobre TAP; sin embargo, hemos de destacar que nuestra principal aspiración es la de aportar GoS a una serie de conexiones privilegiadas, por lo que, en cierto modo, nos alejamos del tratamiento igualitario de las fuentes. No obstante, para soportar esta posibilidad y evitar el comportamiento inequitativo de las conexiones privilegiadas con respecto a las que no lo son, hemos decidido aplicar un mecanismo de atención de las colas basado en pesos. Los pesos de las conexiones están basados en dos parámetros importantes como son el PCR de cada fuente y también en la longitud de cada una de las colas de entrada. Por tanto, atenderemos las colas de entrada, no sólo en función de la velocidad pico de las fuentes, sino también en función del tamaño de todas las colas de espera del resto de fuentes. De este modo, conseguimos atender en primer lugar las fuentes con mayores requerimientos, pero a la vez conseguimos evitar la inanición de las fuentes más lentas. En nuestro caso debemos añadir a los mecanismos propuestos en la literatura dos nuevas características como son la atención prioritaria a las retransmisiones provocadas como consecuencia de las congestiones y, además, el soporte de unidades de procesamiento en forma de células o de PDU que se va a disponer en las colas. Esto quiere decir que al basar el mecanismo de justicia en el tamaño de las colas, el tener unas colas con PDU y otras con células individuales, tenemos que aplicar una proporcionalidad entre el parámetro de tamaño de las colas. En nuestro caso particular consideramos que el tamaño de las PDU es de 1.500 octetos, lo que da lugar a 32 células. Por esto, el tamaño de las colas está calculado en relación al valor proporcional entre PDU y células. La adaptación a PDU de más de 1.500 octetos es inmediata y únicamente requiere de la asignación de una cantidad mayor de memoria.

Cada esquema de gestión de tráfico requiere gestión de colas, y los diversos métodos de gestión de colas tienen diferentes efectos en el tráfico que fluye a través de las colas. Sabemos que los sistemas *Work conserving* (FIFO) envían las PDU cuando el conmutador completa el tiempo de servicio. Por tanto, el conmutador no permanecerá en estado de inactividad (*idle*) si hay PDU en cualquiera de las colas. Por otro lado, los esquemas *Non-work conserving* esperan un espacio de tiempo aleatorio antes de servir la siguiente PDU de las colas, incluso si hay PDU esperando en las colas. Mientras las redes de conmutación de paquetes usan control de flujo basado en ventana (FIFO), las redes de alta velocidad necesitan mecanismos basados en tasas de transmisión y usan servicios *work-conserving* y mecanismos como Fair Queueing (FQ). FQ espera *n-1* bits veces antes de enviar y tiene el problema de que cada fuente dispone de la misma fracción de ancho de banda. Sin embargo sabemos que Weighted Fair Queuing (WFQ) aporta una elevada garantía de rendimiento. Los algoritmos diseñados para conseguir asignaciones de ancho de banda justas ofrecen importantes ventajas para el control de congestión, pero su complejidad de implementación (planificación *per-flow*, gestión de buffer *per-flow* y clasificación *per-PDU*) es un obstáculo importante para su aplicación en redes de alta velocidad. En nuestro caso proponemos una variante de WFQ, con coste constante que actúa por delegación sobre el agente WFQA y cuyo funcionamiento describiremos siguiendo la *Figura 10.4*.

Para comprender el funcionamiento del algoritmo QPWFQ que hemos propuesto en el *Capítulo 4*, retomamos la *Figura 4.3* que es reinterpretada en la *Figura 10.4* en la que podemos observar 4 colas de entrada de nuestro modelo de AcTMs, en la cual podemos observar dos conexiones privilegiadas que generan PDU, y otras dos que generan células independientes y que usamos como tráfico de *background*[1] para estudiar el comportamiento del algoritmo QPWFQ que será retomado en el siguiente capítulo. Para comprender mejor el funcionamiento hemos etiquetado con números cada uno de los pasos que se dan en el procesamiento del tráfico.

Seguidamente describimos los apartados más importantes del flujo de datos y control dentro de las colas de entrada que, como observamos en la figura, está controlado por el agente WFQA. En la Figura hemos distinguido con trazos diferentes los datos, del control y de la petición de retransmisión, y hemos etiquetado cada una de las etapas para comentarlas seguidamente (el algoritmo será descrito en el *Capítulo 11*).

① Esta etiqueta intenta destacar el papel jugado por el agente programable CoSA en el tráfico de datos, para que podamos observar que al conmutador llegan los datos en forma de células que el propio agente de CoS se encarga de ensamblar en unidades de PDU antes que pasen a las colas de entrada para ser planificadas hasta el buffer, según lo que describimos a continuación.

② En este punto intentamos representar el punto de comunicación entre los agentes CoSA y WFQA, de forma que el flujo de datos acaba llegando a WFQA a través del propio agente de CoS. Como sabemos ya, cada una de las colas soporta el tráfico de VPI/VCI distintos, por lo que en este apartado se realiza la elección de la cola a la que va a parar cada una de las células en función de los valores de VPI/VCI que éstas traen en sus cabeceras y de los pesos asignados a las fuentes que se asocian también a la correspondiente cola de entrada que es la que soportará todo el tráfico de esa conexión.

③ Cada una de las colas tendrá un punto de acceso, de forma que las células o PDU se incorporan a cada una de sus correspondientes colas de entrada, tal que, una vez introducidas, debe incrementarse el tamaño actual de la cola, ya que el algoritmo que empleamos para la gestión de las colas se basa, no sólo en los pesos de éstas, sino también en su longitud que permite conseguir la justicia de forma que las colas de menor peso sean atendidas a medida que va creciendo su longitud.

④ En este punto se establecen las condiciones de atención de las cabeceras de las colas, para lo cual mantenemos una cola de turnos en la que se almacena el número de la cola que debe ser transferida hasta el buffer. Como podemos comprobar en la *Figura 10.4*, la condición principal de entrada en la cola de turnos es que la longitud de cada cola sea menor o igual que el peso de esa cola, o bien que se trate de una retransmisión que decidimos priorizar sobre el resto de transferencias. Al introducir cada PDU o célula en su correspondiente cola de entrada se comprueban estas dos condiciones, y si se cumplen, es introducido el número de la cola en la cola de turnos[2]. Así se consigue que las conexiones con un mayor PCR (peso) tengan un límite de entrada en la cola de turnos lo que evita que los VPI/VCI con menor velocidad sean relegados a favor de las fuentes más rápidas.

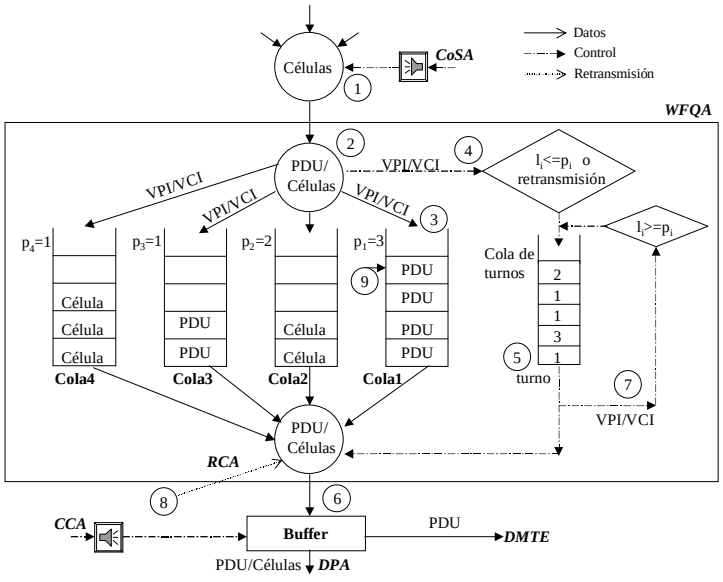

Figura 10.4. *Ejemplo de funcionamiento de las colas de entrada mediante el algoritmo QPWFQ*

[1] Empleamos fuentes de tráfico no privilegiadas adicionales para estudiar el comportamiento de sobrecarga en los conmutadores.

[2] Cuando $l_i > p_i$ se introduce la célula o PDU en su cola, pero no se pasa el número de cola a la cola de turnos.

⑤ En esta etapa se decide el número de la cola cuya cabecera va a ser procesada de modo que será transferida hasta el buffer del conmutador. En el caso de la Figura se indica que la cabecera de la cola número 1 será sacada de la misma, enviada al buffer y su espacio es liberado de la cola 1 y también de la cola de turnos, en la cual la siguiente cola en atenderse será la número 3.

⑥ Este punto indica la operación de transferencia desde las colas hasta el buffer, donde el algoritmo QPWFQ deja el control de la transferencia y pasa al mecanismo que gestiona el propio buffer que se haya indicado desde el agente CCA y que es descrito en apartados siguientes.

⑦ De este modo se garantiza la característica *work conserving* del mecanismo de control de las colas, de forma que, aunque el acceso principal a la cola de turnos especifica que la longitud de cada cola debe ser menor que el peso de esa cola para garantizar justicia, por este punto se permite el acceso a la cola de turnos, aunque la longitud de la cola sea mayor que el peso. Esto permite que, mientras existan paquetes en una cola pendientes de ser procesados, éstos puedan ser procesados sin tener que esperar el turno de otra cola en la que no existen paquetes que tratar.

⑧ Con esta etiqueta identificamos el punto por el cual el agente RCA notifica al WFQA que va a realizar una petición de retransmisión, de forma que, cuando ésta llegue, debe ser priorizada sobre el resto de transferencias pendientes de ser servidas. Este mecanismo de sincronización entre agentes facilita y optimiza la labor de retransmisión. El agente RCA transfiere el índice de la PDU que se va a retransmitir.

⑨ Una vez que el agente WFQA conoce por el agente RCA de la posible llegada de una PDU retransmitida, mantendrá el índice para identificarla rápidamente, y cuando llega la PDU esperada ésta es apuntada de forma específica para pasar a ser la cabecera de su cola correspondiente y ser priorizada con respecto a su cola y con respecto al resto de las colas.

Podemos intuir que el mecanismo que acabamos de describir consigue que los flujos de datos sean etiquetados en función de sus pesos (PCR para nosotros), evita que las fuentes rápidas releguen a las más lentas, garantiza que no se pierda tiempo de procesamiento mientras exista actividad en las colas y, sobre todo, garantiza un acceso constante en la atención de las colas de forma que podemos implementar eficientemente una variante del algoritmo WFQ. Esta sección ha mostrado sólo la forma de atender el tráfico, mientras en el siguiente capítulo tendremos la oportunidad de analizar el algoritmo que implementa esta importante sección de la gestión de las colas de entrada.

10.2.2. BUFFER

En nuestro prototipo de conmutador, el buffer es el punto más susceptible de congestiones, ya que la capacidad de almacenamiento de las colas de entrada es superior a la de éste, y el tráfico de las colas acaba multiplexándose en el buffer de entrada donde pueden acabar produciéndose congestiones en función de la velocidad de transmisión de las fuentes. Por esta causa, hemos decidido aplicar un mecanismo de actuación para detectar las congestiones del buffer, así como otras técnicas para evitar la fragmentación de las PDU y también el *interleaving* de los distintos flujos que acaban confluyendo en el buffer y que tienen el mismo puerto de salida. Como ya sabemos, la política de gestión del buffer es gestionada por el agente CCA que intenta controlar las congestiones aplicando el algoritmo EPDR (o cualquier otro que se desee ya que éste es un agente programable) y también el valor umbral del buffer que es de vital importancia para controlar las congestiones y para evitar la fragmentación de las PDU.

El dimensionado del buffer es una tarea que presenta una gran variedad de facetas que convierten a esta labor en realmente compleja. Así, la elección del tamaño del buffer depende de aspectos como: el número de fuentes simultáneas que vayan a soportarse; la velocidad máxima de cada una de las fuentes en particular y de las características diferentes de cada una de las fuentes de tráfico, de forma que las fuentes a ráfagas hacen bastante inviable la toma de decisión de la dimensión del buffer. Hay que destacar que, cuanto más grande elijamos el buffer, más se tarda en el acceso del tráfico a la red y viceversa. Por esto la elección de la dimensión tiene un relativo impacto sobre la función CAC de control de admisión. En general se asume que las aplicaciones sensibles al retardo requieren buffers de pequeño tamaño, mientras las aplicaciones sensibles a las pérdidas (para las que es adecuada TAP) tendrán mejor comportamiento con buffers de gran tamaño.

La CoS UBR, para la que principalmente se propone TAP, podría ser considerada dentro del tráfico a ráfagas, por lo que podemos enfrentarnos al problema de la elección del tamaño del buffer según los dos planteamientos más tradicionales [3]:

- Multiplexación sobre la velocidad disponible que consiste en restringir el número de fuentes con tráfico a ráfagas para que la velocidad total de entrada no exceda el CSR (Cell Slot Rate) total del enlace, y si lo exceden se supone que todo el exceso de tráfico se perderá.
- Multiplexación estadística compartiendo la velocidad, que asume que se dispone de un buffer de gran tamaño para soportar las células debidas al exceso de velocidad, de forma que sólo se perderá una porción de ellas mientras el resto de las que no se pierden experimentan retardos en el buffer.

En el caso de emplear un buffer pequeño, y apoyándonos en un modelo finito de cola M/D/1, puede calcularse el retardo máximo en la cola multiplicando la capacidad del buffer por el tiempo por slot de células, *s* en la velocidad apropiada del enlace. El retardo medio *d* en la cola *q* depende de la carga, ρ y es calculada usando la siguiente fórmula para un sistema infinito M/D/1:

$$d_q = s + \frac{\rho s}{2(1-\rho)} \tag{1}$$

De aquí puede deducirse que, para un sistema M/D/1 finito, el retardo medio puede quedar bastante ajustado ya que las pérdidas son muy bajas; es decir, los retardos sólo tendrán una diferencia apreciable en el caso que las pérdidas en el sistema finito sean muy elevadas, lo cual es bastante improbable. La teoría de colas demuestra que el retardo sólo incrementa a medida que nos acercamos al 80% de carga. Esto nos permite determinar que los buffers de pequeño tamaño son aplicables a redes que ofrecen la capacidad de transferencia de DBR (Deterministic Bit-Rate) y la capacidad de transferencia SBR (Statistical Bit-Rate) basadas en lo que antes hemos denominado como multiplexación sobre velocidad disponible.

En el caso de emplear buffers de gran tamaño para el tráfico a ráfagas, el dimensionado es mucho más complicado que en el caso anterior porque intervienen muchos más parámetros de caracterización de tráfico. En este caso suele asumirse un tráfico medio de muchas fuentes ON/OFF, cada una de ellas con las mismas características de tráfico (PCR, MCR y MBL en el estado activo). Los parámetros clave suelen ser el mínimo número de velocidades pico requeridas por ráfaga de encolado, N_0, el ratio de la capacidad del buffer para la longitud media de ráfaga, *X/b*, la carga media ρ, y la probabilidad de pérdida de células *CLP*. Por tanto, es necesario calcular el valor de *X/b* que es encontrado resolviendo la ecuación:

$$\frac{CLP_{fuente}}{CLP_{bsl}} = CLP_{bsd} = \exp\left[-N_0 \frac{X}{b} \frac{(1-\rho)^3}{4\rho+1}\right] \tag{2}$$

donde *bsl* es el factor *burst scale loss* y *bsd* el factor *burst scale delay*. Adaptando debidamente la fórmula anterior obtenemos la siguiente expresión para poder calcular la dimensión del buffer que necesitamos,

$$\frac{X}{b} = -\frac{4\rho+1}{(1-\rho)^3} \frac{\ln(CLP_{fuente} / CLP_{bsl})}{N_0} \tag{3}$$

En el caso de TAP vamos a requerir un buffer de gran tamaño, por las propias características del tráfico. Como acabamos de comprobar, es realmente complejo ajustar un tamaño acertado por la dificultad de caracterizar todos los parámetros del tráfico. Por esto, en nuestro caso particular, y apoyándonos en las consideraciones expuestas, hemos fijado el tamaño del buffer en 2.000 células. Este es un valor ampliamente usado en la literatura que permite equilibrar el efecto entre retardo en los accesos y la probabilidad de pérdida provocada por la velocidad de transferencia de las fuentes y por el número de fuentes que pueden acabar congestionándolo. En nuestro caso, la elección de 2.000 células como tamaño de buffer supone una zona de memoria de 106.000 octetos, lo que, usando PDU de 1.500 octetos, acaba dando que en el buffer pueden tener cabida un total de 70 PDU almacenadas en espera de ser servidas a sus correspondientes puertos de salida. Consideramos este valor como aceptable para poder obtener del algoritmo EPDR un mejor rendimiento, con una reserva de memoria razonable y asumible por cualquier conmutador en la actualidad.

10.2.3. MEMORIA DMTE

La memoria DMTE (Dynamic Memory Trusted EAAL-5 PDU) puede ser considerada como el módulo clave de la arquitectura TAP. Se comporta como una zona de memoria compartida en la que se copian las PDU en los conmutadores activos. Cuando una PDU de una conexión privilegiada llega al buffer, antes de ser aplicada la técnica del VC Merge es realizada una copia en la memoria DMTE. Por tanto, DMTE lo que hace es almacenar temporalmente las PDU generadas por la extensión de AAL-5 que hemos propuesto (EAAL-5) para la arquitectura. Una vez realizada la copia de cada PDU, éstas son enviadas a sus correspondientes puertos de salida. De este modo, estas copias de PDU contenidas en la DMTE pueden ser usadas para atender las peticiones de retransmisión en el caso que se produzcan congestiones en los conmutadores siguientes al AcTMs que ha realizado la copia.

Como podemos ver, por tanto, la DMTE es la que aporta al protocolo TAP la característica *trusted* que hemos justificado en el *Capítulo 3* y que acaba dando como resultado la GoS. Por la forma en que hemos diseñado la memoria, ésta permite almacenar varias PDU de cada conexión privilegiada, con la intención de aportar una mayor garantía de servicio en función de la cantidad de PDU que se almacenan de cada conexión. De todos modos, la propia característica de las PDU no nos permite disponer de un gran número de unidades de PDU en la memoria, porque, a medida que crece el número de conexiones fiables, el tamaño de la DMTE crece también considerablemente. Para evitar el crecimiento incontrolado del índice de ocupación de la memoria aplicamos un control sobre la misma de forma que las PDU de cada conexión permanecen en la memoria, sólo si se dispone de suficiente espacio de almacenamiento en la misma. Es decir, cuando se requiere almacenar una nueva PDU, se aplica un mecanismo de *aging* que permite sacar de la DMTE aquellas PDU de una conexión que llevan más tiempo en la memoria. Esta posibilidad es la que aporta la característica Dinámica a la memoria DMTE.

Debemos destacar que las PDU de AAL-5 pueden tener un elevado tamaña (hasta 65.535 bytes) y, dado el potencialmente alto número de conexiones (VPI/VCI) con requerimientos de GoS, el tamaño de la DMTE puede ser excesivo. Esta es la razón por la que hemos limitado el número de PDU que se pueden almacenar por cada conexión fiable como hemos comentado en el párrafo anterior. De este modo, sólo soportamos un número reducido de VCI privilegiados con transferencias garantizadas. Es evidente que el tráfico de una conexión es mucho más seguro cuando la DMTE almacena más PDU de esta conexión. Pero el tamaño de la DMTE también depende del tamaño de las PDU que, como sabemos, es variable y de considerable tamaño en su valor máximo como hemos visto antes. Por esta causa hemos calculado el tamaño que requerimos de memoria para poder garantizar un número concreto de conexiones con mayor o menor grado de confianza en cuanto a la GoS.

Podemos entender que el tamaño del la memoria DMTE es un importante factor en la arquitectura. Así, trabajamos con dos parámetros para dimensionar esta memoria que son el número de PDU almacenadas de cada conexión y, por otro lado, el tamaño de cada una de esas PDU. Se estudian a continuación las combinaciones de estos dos parámetros.

En el caso de usar las PDU de EAAL-5 con tamaño máximo requeriremos los siguientes tamaños de memoria si decidamos mantener en la DMTE 3 PDU almacenadas por cada conexión privilegiada:

3 PDU * 64 Kbytes cada PDU = 192 Kbytes * 10 conexiones con GoS = 1,9 Mbytes de DMTE.

3 PDU * 64 Kbytes cada PDU = 192 Kbytes * 20 conexiones con GoS = 3,8 Mbytes de DMTE.

Si consideramos un tamaño de PDU más razonable con 1.500 bytes, como es lo más lógico para el tráfico TCP que es una de nuestras motivaciones generales, tendremos los siguientes requerimientos de tamaño en DMTE:

3 PDU * 1,5 Kbytes cada PDU = 4,5 Kbytes * 10 conexiones con GoS = 45 Kbytes.

3 PDU * 1,5 Kbytes cada PDU = 4,5 Kbytes * 100 conexiones con GoS = 450 Kbytes.

3 PDU * 1,5 Kbytes cada PDU = 4,5 Kbytes * 1.000 conexiones con GoS = 4,5 Mbytes.

La *Tabla 10.1* muestra esquemáticamente algunos de los resultados obtenidos, donde podemos observar los resultados requeridos de memoria en función del número de conexiones que se desea privilegiar y el número de PDU de cada conexión a fiabilizar. Podemos ver así cómo aportar GoS a 1.000 fuentes de tráfico TCP sólo requiere de una memoria de 3 Mbytes que podemos considerar como un tamaño más que razononable y asumible en los conmutadores actuales.

TABLA 10.1
TAMAÑO REQUERIDO DE DMTE PARA OFRECER GOS A X CONEXIONES

Nº de PDU almacenadas por cada conexión	Tamaño de cada PDU	Nº de Conexiones con GoS	Tamaño de DMTE
3	64 Kbytes	10	1.9 Mbytes
3	64 Kbytes	20	3.8 Mbytes
3	1,5 Kbytes	10	45 Kbytes
3	1,5 Kbytes	100	450 Kbytes
3	1,5 Kbytes	1.000	4,5 Mbytes
4	1,5 Kbytes	1.000	6 Mbytes
2	1,5 Kbytes	1.000	3 Mbytes

Podemos comprobar por tanto cómo el grado de GoS que van a tener las fuentes privilegiadas depende del número de PDU que se almacenan en la DMTE. Vemos también cómo el grado de confianza deseado afecta también al tamaño de memoria consumida. Para nuestros planteamientos partimos de un tamaño de DMTE fijo, con el que se equipa a cada conmutador AcTMs. Así, con un tamaño fijo podemos determinar el número total de fuentes privilegiadas que podemos garantizar, o bien el grado de fiabilidad que desea aportarse. Debemos, por tanto, buscar un punto de equilibrio entre los tres factores: Tamaño de PDU, Número de fuentes con GoS y grado de fiabilidad deseado para cada fuente que depende del número de PDU de cada conexión que se almacena en la DMTE. Si acordamos que el grado de GoS (G) depende del tamaño de memoria DMTE con que equipemos a un conmutador podremos expresar este grado de la siguiente forma,

$$G = \frac{T}{\sum_{1}^{n} N * t} \tag{4}$$

donde T es el tamaño total de la memoria DMTE, n es el número de fuentes, N es el número de PDU que se almacena de cada fuente o VCI y t es el tamaño de cada una de esas PDU. Así, en el último ejemplo de la *Tabla 10.1* el grado de GoS G es 1, ya que T=3 Mbytes; tenemos 1.000 fuentes de las cuales se almacenan 2 PDU, de 1.500 octetos cada una. En cierto modo, y obviando otras características de las fuentes y del conmutador, el valor de G debería tener un valor mayor o igual a 1 para poder ofrecer a un número n de VCI su GoS. O dicho de otro modo, y si expresamos la fórmula (4) de otra forma, podemos obtener el grado de garantía de servicio g de la fuente i o VCI i en función de el número de PDU que deseamos almacenar y del tamaño de cada una de esas PDU,

$$g_i = \frac{T}{N * t} = \frac{3Mbytes}{2 * 1.500bytes} = 1.000 \tag{5}$$

La expresión (5) está calculada nuevamente sobre el último ejemplo de la *Tabla 10.1*, y podemos ver cómo el valor obtenido para la g coincide con el número de fuentes de la característica de la fuente i que podrían soportarse con una memoria DMTE de 3 Mbytes. Llamamos la atención sobre el hecho de que el parámetro g para las fuentes individuales ha de ser normalizado o entendido como que su valor ideal sería 1, ya que a mayores valores de N (y también de t) dispondremos de mayor GoS aunque el valor de g_i obtenga valores menores. Por tanto, mientras en el caso general de G lo deseable es obtener valores positivos cercanos a 1, en el caso de g_i lo razonable es obtener también valores positivos pero bastante alejados de la unidad, ya que en realidad lo que expresa es el número de VCI de iguales características que podrían soportarse con ese valor de g_i.

Los accesos a la DMTE son otro aspecto de importancia ya que debemos disponer de un mecanismo óptimo para poder localizar de forma eficiente las PDU en las solicitudes de retransmisión. Como podemos observar en la *Figura 10.2*, la copia de cada PDU es realizada en la memoria DMTE desde el buffer, justo antes de ser enviada a su correspondiente puerto de salida. Para poder localizar y acceder con rapidez a las PDU de la DMTE hemos decidido utilizar un índice que está formado por los valores *VPI/VCI/PDUid/Port_Entrada*. Con estos valores mantenemos la DMTE como una tabla de *hash* cuya función de *hashing* elimina las posibles colisiones en los accesos de escritura y lectura.

Por tanto, cada PDU dentro de la memoria es identificada por el VPI/VCI de la conexión a la que pertenece, su propio identificador de PDUid (que se corresponde con los campos UU y CPI de AAL-5 como ya sabemos) que las distingue con respecto al resto de PDU de su misma conexión y, además, hemos introducido el valor del puerto de entrada para distinguir la posibilidad de que a un mismo conmutador

accedan conexiones diferentes en las que se hayan elegido valores idénticos de VPI/VCI por provenir de usuarios diferentes que, en principio, no tienen porqué conocer los valores de identificación de las conexiones de otros usuarios. Esta probabilidad de coincidencia es relativamente alta, ya que existe la tendencia a elegir valores de VPI/VCI similares en el establecimiento de la conexión, por lo que hemos decidido pagar este precio para evitar la posibilidad de claves repetidas en la función de *hash* de la DMTE. Además de las colisiones en la función de *hash* también esto daría lugar a retransmisiones erróneas, de forma que se podrían retransmitir PDU de una conexión a otra por esa posibilidad de repetición en el índice.

La operación de escritura en la memoria es la que se encarga también de controlar el número de PDU que se ha decidido almacenar por cada una de las conexiones, por lo que aplica la citada técnica de *aging*, mediante la cual se evita (cuando no existe suficiente espacio de memoria) que una PDU entrante no tenga cabida. Es decir, mientras existe espacio libre de memoria se va a permitir la entrada de PDU, de forma que cuando el espacio comienza a ser insuficiente es cuando se eliminan las PDU cuyo PDUid es menor (bajo la suposición que son las que mayor tiempo llevan en la memoria) y por lo tanto son las menos susceptibles de requerir una retransmisión. Una situación extrema que puede describirse es la que se produce cuando al retransmitir una PDU desde la DMTE llega una nueva PDU desde el buffer con el mismo VPI/VCI y no existe espacio libre en la memoria, en este caso lo que se hace es descartar la PDU entrante de forma similar a la pérdida que se puede producir cuando se carece de buffers en *VC Merge*.

Lo razonable en los conmutadores activos es disponer de un espacio fijo y concreto de memoria que será repartido entre todos los VCI que se deseen fiables. Mediante un parámetro de tráfico puede expresarse el número de PDU que se desea almacenar de cada VCI, de forma que mediante el sistema multiagente que explicaremos en secciones siguientes, se pueda controlar este valor en los accesos a la memoria DMTE de cada uno de los conmutadores AcTMs. Es decir, el agente CoSA soporta un parámetro de tráfico que es el del grado de GoS deseado por la fuente, tal que el valor normalizado de la g_i expresa en cada conmutador la reserva máxima de DMTE que debe hacerse para cada VCI. Destacamos también que en nuestras simulaciones podemos determinar el tamaño deseado de DMTE, aunque este valor es el mismo en todos los conmutadores de la red. En el *Capítulo 12* tendremos ocasión de explicar las características del simulador de TAP, donde veremos cómo éste nos permite elegir, tanto los tamaños de DMTE, como el número de PDU y el tamaño de cada una de ellas. El objetivo final es el de conseguir el mayor número posible de conexiones garantizadas, o bien el de lograr el mayor grado posible de GoS para un número más reducido de fuentes. El simulador nos permitirá demostrar que es aceptable el tamaño de memoria que hay que aportar a los conmutadores para conseguir un aspecto tan importante como es el de las transferencias garantizadas que la propia tecnología ATM no es capaz de soportar con los actuales estándares.

10.2.4. TABLAS DE E/S

Como sabemos, a cada puerto de un conmutador AcTMs le corresponde una Tabla de Entrada/Salida que almacena los índices de acceso a la memoria DMTE. Estos índices nos permiten un acceso eficiente a través de la función de *hash* de la memoria DMTE en el caso de solicitudes de retransmisión. Pero, además, estas tablas desempeñan otra importante función que es la de evitar la repetición de valores VPI/VCI en conexiones diferentes. Como es conocido, en el establecimiento de la conexión los usuarios de ATM pueden especificar el VPI/VCI que desean usar en sus conexiones, por esto es posible que en un conmutador confluyan dos conexiones que provienen de usuarios diferentes que han decidido usar el mismo valor de VPI/VCI. Debemos buscar entonces un sistema que nos permita identificar las PDU de cada emisor que se han almacenado en la DMTE, ya que las dos conexiones pueden coincidir, no solo en VPI/VCI, sino también en el identificador de PDU, PDUid, y también en el puerto de entrada que son los valores que configuran el índice de acceso a la DMTE. Como los VPI/VCI también pueden cambiar de la entrada a la salida el sistema adoptado debe permitir identificar la secuencia para poder localizar las PDU en las retransmisiones.

El sistema de identificación de las PDU implementado se basa en el uso de las Tablas de E/S asociadas a cada uno de los puertos de salida de los conmutadores que permitan relacionar los valores de entrada de cada PDU con los valores de salida una vez que han sido conmutadas a la salida. Es decir, cuando una PDU es segmentada y enviada a su puerto de salida, en la Tabla de E/S de este puerto se almacena el siguiente índice,

InPort/VPIIn/VCIIn/PDUid/OutPort/VPIOut/VCIOut/OutPortPrev/InPortNext

El significado de cada uno de los campos que componen el índice de las tablas se explica a continuación:

- *InPort*: Representa el valor numérico del puerto de entrada de las PDU de cada una de las conexiones privilegiadas. Normalmente será un valor comprendido entre 1 y 16 (en el caso de conmutadores con 16 puertos) de forma que identifica la conexión física de cada conmutador por la que se va a realizar el transporte de los datos entre conmutadores.

- *VPIIn*: Este valor representa el identificador de camino virtual (VPI) con el que cada una de las conexiones privilegiadas llega a un conmutador. Este valor estará comprendido entre 0 y 255 en el caso de los nodos UNI, y entre 0 y 4.095 en el caso NNI como puede deducirse de la *Figura 1.1*. Este valor está íntimamente ligado al valor del puerto de entrada *InPort* anterior, ya que ésta es la vía física por la que el camino virtual se va a establecer en cada conmutador.

- *VCIIn*: Este campo identifica el circuito virtual (VCI) por el que se realiza el envío de las células entre conmutadores. Este valor está comprendido entre 0 y 65.535, tanto en los nodos UNI como NNI, de forma que cada VPI puede tener asociados, como máximo, hasta los 65.535 VCI por los que se pueden realizar otras tantas conexiones. Como es sabido, la conmutación de un VPI concreto permite la conmutación automática de todos los VCI que lleva asociados. El *VCIIn* indica entonces el valor con el que las células acceden a cada conmutador, dentro de un VPI y a través de un *InPort* concreto.

- *PDUid*: Este es el campo que identifica cada una de las PDU de una determinada conexión con GoS. Como ya sabemos, este valor toma valores entre 0 y 65.535 según se explica en la *Figura 9.1* y se corresponde con los campos UU y CPI de un octeto cada uno. Este campo permanecerá invariable e-e, por lo que no cambiará de valor al atravesar los conmutadores de la red. En realidad este es el campo más importante para poder identificar las PDU a retransmitir, pero sin los tres campos anteriores no será posible identificar PDU con mismo valor de PDUid pero pertenecientes a diferentes conexiones *Port/VPI/VCI*.

- *OutPort*: Con este valor se identifica el puerto de salida por el que las células de entrada de una conexión concreta con valores *InPort/VPI/VCI* acabarán llegando al siguiente conmutador en sentido hacia el destino. Puede tomar el mismo rango de valores que el *InPort* ya comentado.

- *VPIOut*: Identifica el valor del VPI de salida que la matriz de conmutación asignará a las PDU de una determinada conexión. Este valor, como el *VPIIn*, se asignará a cada una de las cabeceras de las células en las que se segmente cada PDU, con la característica que ese será el valor con el que llegarán las células al siguiente conmutador, de forma que el valor de *VPIOut* en el conmutador *n* será el mismo que el *VPIIn* en el conmutador *n+1*. El rango de valores que puede tomar es el mismo que en el caso de las entradas.

- *VCIOut*: Su función es la misma que la del campo anterior, sólo que en este caso se trata del valor del VCI de salida. Sus valores potenciales se corresponden con los del *VCIIn*, con la característica que ahora el valor del *VCIOut* del conmutador *n* coincidirá con el valor *VCIIn* del conmutador *n+1*. Tenemos por tanto ya la relación de los valores de entrada con los de salida; es decir, cada identificador de PDU fiable PDUid, llegará a un conmutador activo a través del camino indicado por la terna */InPort/VPIIn/VCIIn*[3] y será servido al siguiente conmutador a través del camino representado por */OutPort/VPIOut/VCIOut*.

- *OutPortPrev*: Este campo del índice de acceso de las Tablas de E/S identifica el número de puerto de salida del conmutador previo al actual por el cual salieron las células de una PDU para llegar hasta el conmutador actual. Es decir, si estamos en el conmutador *n*, el valor de *OutPortPrev* es el valor del puerto de salida del conmutador *n-1* (previo al *n*) desde el que se han enviado las células al *InPort* o puerto de entrada del conmutador *n*.

- *InPortNext*: Es similar al campo anterior, sólo que en este caso este valor indica el número de puerto de entrada por el que se va a acceder al siguiente conmutador. Es decir, si estamos en el conmutador *n*, el *InPortNext* indica el puerto de entrada por el que el conmutador *n+1* (siguiente a *n*) va a recibir las células de una PDU fiable y que el conmutador *n+1* identificará como el *InPort*. Estos dos últimos campos del índice son necesarios para que cada conmutador conozca la relación entre los números de los puertos de entrada y salida con sus conmutadores vecinos. Es decir, para nuestros objetivos necesitamos saber la correspondencia de cada uno de los puertos de salida del conmutador *n* con cada uno de los puertos de entrada del conmutador *n+1*. Del mismo modo, es necesario conocer la relación de cada uno de los puertos de entrada del conmutador *n* con los puertos de salida del conmutador *n-1*. Para ello requeriremos actuar sobre alguno de los aspectos de la señalización en la red, ya que es vital disponer de estas relaciones para poder resolver las solicitudes de retransmisión.

Podemos observar en la *Figura 10.5* la situación de las Tablas de E/S del puerto 6 con cada uno de los campos explicados. Observamos también la relación entre el índice de la Tabla con el índice de la DMTE, de forma que sólo coinciden en los valores de entrada de la PDU, dejando los valores de salida para relacionarse con el conmutador siguiente.

[3] Destacamos que el valor formado por /InPort/VPIIn/VCIIn/PDUid constituye el índice de acceso a la DMTE.

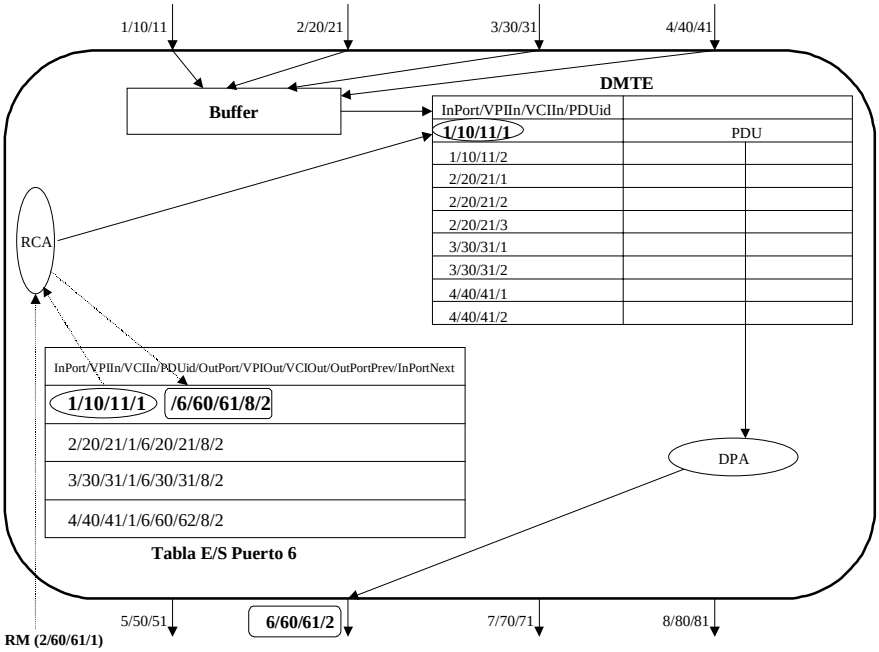

Figura 10.5. *Contenidos de las Tablas de Entrada/Salida de los AcTMs*

Como puede observarse en la *Figura 10.5*, a cada uno de los puertos de entrada del conmutador se ha asignado un VPI/VCI, de forma que, por ejemplo, por el puerto 1 accede la conexión cuyo VPI/CVI es el 10/11. Las PDU van llegando a la DMTE a través del buffer de modo que cada PDU del mismo VPI/VCI va aumentando en su número de secuencia. En la Tabla E/S del puerto 6 puede verse cómo la PDU 1 que proviene de la conexión VPI/VCI=10/11 a través del puerto 1, cuyo índice en la DMTE es 1/10/11/1, tiene en la Tabla el mismo comienzo de índice. Pero esta PDU va a salir por el puerto 6 asignando como VPI/VCI de salida el 60/61. El puerto por el que va a entrar en el siguiente conmutador es el 2 y el puerto de salida del conmutador anterior por el que llega al de la figura es el 8.

La *Figura 10.5* muestra también cómo desde el conmutador siguiente al representado se está pidiendo retransmitir, mediante una célula BRM, la PDU cuya identificación en ese conmutador es 2/60/61/1, es decir, ha entrado por el puerto 2, su VPI/VCI es el 60/61 y el PDUid es el número 1. Este índice parcial es recibido por el agente RCA que se encarga de usarlo para acceder a la Tabla en la que se localiza la PDU 1 cuyos datos de entrada son 1/10/11/1, que es el índice devuelto al agente RCA y que éste empleará para acceder a la DMTE y localizar los datos de la PDU que será retransmitida nuevamente por el agente DPA.

El mecanismo ideado es mantenido por el agente DPA que, antes de despachar las células de una PDU, se encarga de añadir al índice que se dispone de la entrada los nuevos valores de la salida, dando lugar a un nuevo índice que será empleado cuando a este conmutador llegue una célula BRM con el índice de retransmisión. Éste es el que se empleará como acceso a la Tabla de E/S para encontrar el valor del índice que nos sirve para poder luego acceder a la DMTE a localizar los datos de la PDU que se debe retransmitir.

Podemos observar que en realidad estas tablas de E/S se comportan a modo de tablas de *routing* ya que se encargan de mantener la relación de los datos con que entra cada PDU en los conmutadores con los datos de salida del mismo conmutador. Mediante esta relación podrá posteriormente encontrarse el índice de acceso a la memoria de retransmisiones y evitarse un problema más grave que la propia pérdida de una PDU que es la retransmisión de una PDU equivocada o perteneciente a una conexión diferente. Para analizar el comportamiento y explicar su funcionamiento disponemos de la *Figura 10.6* en la que presentamos una traza de los valores que toman las tablas de E/S en varias conexiones, empleando en este caso tres conmutadores AcTMs consecutivos que hemos etiquetado para explicar a continuación su funcionamiento.

① Podemos observar en esta etiqueta cómo en la Tabla E/S del puerto 6 se mantiene la relación de la información de las PDU a la entrada de cada conmutador en la DMTE con lo que se tiene a la salida del conmutador. Por ejemplo, por el puerto 1 VPI/VCI=10/11 recibimos la PDU 1 de esta conexión que después saldrá del conmutador por el puerto 6 con el VPI/VCI=60/61, y acabará llegando al siguiente conmutador por su puerto 2. Suponemos que esta PDU proviene del puerto 8 de salida del conmutador anterior. En la tabla lateral de la derecha anotamos estas relaciones, donde la PDU destacada en negrita ha experimentado una congestión en el buffer del segundo conmutador.

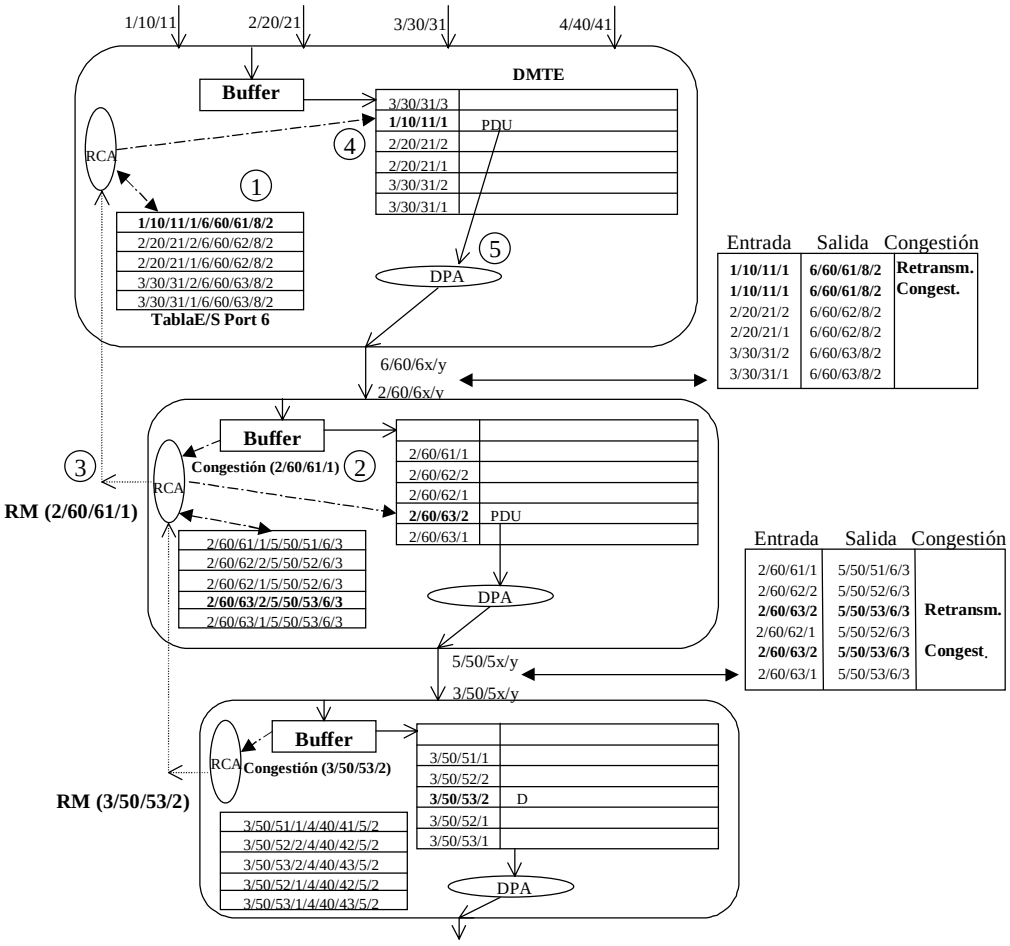

Figura 10.6. *Escenario para analizar las Tablas de E/S en tres AcTMs*

② En este punto es donde suponemos la congestión del buffer del segundo conmutador con la PDU de índice de entrada 2/60/61/1. Podemos ver cómo en el paso del primer conmutador 1 al 2 se pasa desde el puerto 6 en el primero al puerto 2 del segundo, manteniéndose los valores VPI/VCI asignados en el primer conmutador.

③ Al detectarse la congestión del buffer el agente RCA genera la célula BRM con los datos de la PDU, que acaban llegando al primer conmutador a través de su RCA local que localiza la relación entre los datos de entrada y de salida de la PDU según hemos explicado anteriormente.

④ Con el índice de acceso a la DMTE se toma la PDU para retransmitirla. Destacamos que el mecanismo de retransmisión lleva de forma inherente el desorden de las PDU, por lo que al retransmitir una PDU no se puede garantizar el orden en la secuencia. Esta situación se produce en el tercer conmutador que solicitada una PDU al segundo conmutador y que es localizada en la DMTE local de este conmutador. La PDU retransmitida acabará provocando que la PDU retransmitida llegue después que otra PDU que en el origen estaba delante de ella. Hemos marcado esta posibilidad de desorden con una D en la DMTE del tercer conmutador.

⑤ El agente DPA se encarga de la retransmisión de la PDU solicitada hasta su correspondiente puerto de salida y VPI/VCI, según los datos anteriormente empleados en la primera transmisión.

El escenario anterior lo podemos ampliar con la inclusión en la red de conmutadores no activos, que no implementan el protocolo TAP y que, por tanto, no disponen de ninguna de las características hardware y software de los conmutadores AcTMs. En este caso los conmutadores no activos actúan de la forma tradicional, conmutando las células y PDU en dirección al destino, y las BRM en sentido contrario al flujo de datos hasta encontrar un conmutador activo. Este comportamiento provocaría que se perdiese la relación de los puertos de E/S y VPI/VCI en el caso de las solicitudes de retransmisión. Para controlar esta situación lo que se hace en el proceso de establecimiento de la conexión es registrar en todas las tablas de E/S de los

conmutadores activos los VPI/VCI junto a los puertos de E/S de cada uno de los conmutadores no activos que forman parte de la ruta seguida por cada fuente con requerimientos de GoS. Esta labor de registro de la ruta es realizada por el agente CoSA en el establecimiento de la conexión y, como hemos comentado anteriormente, supone actuar sobre los mecanismos de señalización de la red. Todo esto nos da una idea más definida de que las tablas de E/S de los AcTMs actúan a modo de tablas de routing.

En lo relativo a los costes de acceso a las tablas hemos de destacar su acceso constante tanto para lectura como escritura, y su dimensión la realizamos en función del tamaño de la DMTE. Es decir, cada una de las tablas mantiene tantas posiciones como números de entrada tiene la memoria dinámica con la diferencia con respecto a ésta en que las Tablas de E/S sólo almacenan los índices y no los datos reales que son registrados únicamente en la DMTE.

10.2.5. SISTEMA MULTIAGENTE

Como sabemos, la arquitectura TAP dispone de un subsistema constituido por cinco agentes software que hemos dado en denominar SMA-TAP y que es el que aporta la vertiente software a los conmutadores AcTMs y que confiere a estos su característica activa. Como hemos podido comprobar en apartados anteriores de la arquitectura, la mayor parte de bloques hardware son gestionados por alguno de los agentes software que forman el sistema multiagente SMA-TAP. En esta línea vamos a describir la funcionalidad de cada uno de los agentes de la arquitectura en relación a los aspectos de la arquitectura que controlan y de las interacciones entre cada uno de ellos.

La *Figura 10.7* representa la relación entre la arquitectura del SMA-TAP y el modelo arquitectónico de ATM presentado en la *Figura 10.1*. Como podemos observar en la *Figura 10.7* los agentes que intervienen en la arquitectura dependen directamente del protocolo TAP que, en realidad, es el conjunto de algoritmos que se ejecutan sobre la arquitectura TAP. Podemos observar los cinco agentes software, donde dos de ellos son programables y algunos de ellos están coordinados entre sí para poder realizar las labores que tienen encomendadas. Seguidamente vamos a comentar algunos de los aspectos de interés del proceso de comunicación entre los agentes, así como los aspectos de mayor interés de SMA-TAP y, para ello, vamos a emplear las etiquetas colocadas en la *Figura 10.7*.

① El protocolo TAP es el que concentra toda la labor del conjunto de algoritmos que ejecutan cada uno de los agentes software del SMA-TAP. Estos algoritmos que constituyen el protocolo serán descritos en el *Capítulo 11*, pero debemos de destacar que el protocolo TAP debe ser soportado por los nodos extremos de la comunicación de la VPN. Gracias a la característica distribuida de la arquitectura se permitirá el soporte de las conexiones privilegiadas desde el nodo emisor de la conexión y, a través de los conmutadores activos que existen en la VPN, se establecerá la comunicación con el nodo extremo de la comunicación que debe implementar también TAP. Podemos observar cómo el protocolo TAP está situado por encima de la capa EAAL-5 y, a su vez, puede tener por encima los protocolos de capas superiores como TCP, Frame Relay, etc.

② El agente CoSA (Class of Service Agent), situado en la Capa de Plano de Control del modelo arquitectónico ATM, se encarga de identificar las características del tráfico de entrada en la red (PCR, Ton, Toff, etc.), de forma que se responsabiliza del establecimiento de la conexión creando el VPI/VCI para cada una de las fuentes de datos. Este agente programable también se comunica con el resto de agentes CoSA de los conmutadores activos con el fin de mantener las relaciones de caracterización del tráfico en todos los AcTMs que intervienen en la red extremo-extremo. Podemos observar en la *Figura 10.7* que este agente también se comunica con el agente WFQA local a cada uno de los conmutadores, con la función ya explicada en secciones precedentes. CoSA se encarga de reensamblar cada una de las células de las conexiones fiables para constituir las PDU que serán pasadas a las colas de entrada y servidas a las colas ya en unidades de PDU con el objetivo de poder identificar cada una de las PDU de forma individualizada. Para que esto sea posible, los conmutadores deben soportar también la extensión EAAL-5 que hemos explicado en puntos anteriores.

③ El agente programable CCA (Control de Congestion Agent) tiene como función principal la de definir el algoritmo de control de congestión que se establece en el buffer que, en nuestro caso, se trata de EPDR. De este modo, se encarga también de aplicar el valor del umbral necesario para aplicar EPDR sobre el buffer y evitar las fragmentaciones de las PDU. Este agente mantiene también coordinación con el resto de agentes CCA de los conmutadores activos de la red, ya que la labor de programación establecida debe ser común en todos los conmutadores que intervienen en las conexiones garantizadas. Queda clara por tanto la relación del agente software CCA con el buffer como bloque

hardware de los conmutadores. Podemos observar cómo este agente está situado en la Capa de Plano de Gestión de la arquitectura ATM.

④ El agente WFQA, situado también sobre la Capa de Plano de Gestión, se responsabiliza de la gestión de las colas de entrada y, para ello, implementa el algoritmo QPWFQ. Hemos descrito ya las labores que realiza el agente sobre las colas en la sección dedicada a este componente hardware de la arquitectura, y el algoritmo será también comentado en detalle en el *Capítulo 11*, pero destacamos en este punto que este agente juega un papel primordial en la optimización del procesamiento del tráfico de entrada, ya que de él depende el garantizar que se cumplen los pesos aplicados a las fuentes, a la vez que se garantiza la justicia en todas ellas. Destaca también su función en la atención de las retransmisiones de las PDU congestionadas en el mismo o en conmutadores que están por debajo y en dirección al destino. Su coordinación es también importante para conseguir servir las PDU y células al buffer con eficiencia.

⑤ DPA es un agente que actúa desde la Capa de Plano de Gestión, de forma autónoma con respecto al resto de agentes del SMA-TAP. Dispone de funciones que le permiten solicitar al buffer la siguiente unidad de transferencia, tanto desde el buffer en el caso de las transmisiones, como desde la DMTE en el caso de las retransmisiones que hayan tenido éxito en el conmutador local. Para ello debe sincronizarse con estos dos bloques hardware para aprovechar las ventajas de una operación atómica que permite que, en el caso de las PDU, éstas sean transferidas de forma íntegra a su correspondiente puerto de salida para evitar el conocido problema del *interleaving*. Es también función del agente DPA la asignar los nuevos valores de VPI/VCI antes de ser enviadas a la salida las células que la red es capaz de transportar. Antes de comenzar a enviar las células de una PDU, DPA se encarga también de actualizar la correspondiente tabla de E/S según lo que hemos indicado en la sección anterior, para poder tener en el índice el valor del puerto de salida de cada una de las células de una PDU perteneciente a una conexión garantizada.

⑥ Por último, el agente RCA (Retransmission Control Agent) actúa desde la Capa de Plano de Gestión de la arquitectura, realizando una de las labores más importantes en el mecanismo de recuperación de las PDU congestionadas. Este agente recibe notificaciones de congestión desde el buffer local, que se encargar de proporcionarle el índice de la PDU que ha sido descartada del buffer por no tener cabida. Con este índice RCA se encarga de coordinarse con WFQA para notificarle a éste que va a generar una célula RM solicitando la retransmisión de esa PDU al conmutador anterior. El agente puede recibir también notificación de retransmisión desde el conmutador que le sucede, de forma que mediante el índice recibido en una célula RM se encargará de localizarla en la DMTE según detallaremos más adelante. RCA, por tanto, tiene acceso a los bloques hardware DMTE, Tablas de E/S y al buffer. Para realizar su trabajo necesita coordinarse con su agente local WFQA y con los agentes RCA de otros conmutadores a través del VPI/VCI dedicado a las células RM.

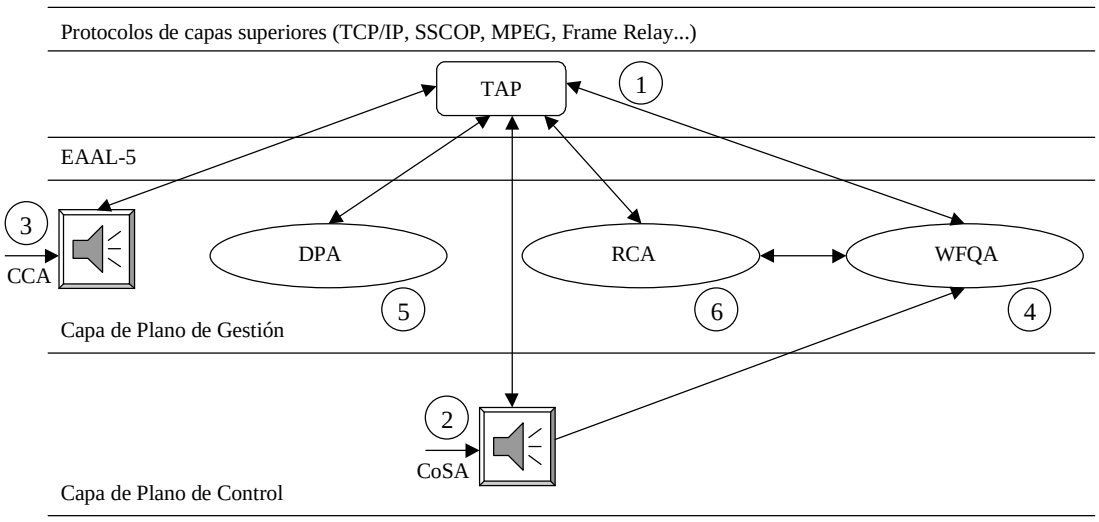

Figura 10.7. *Arquitectura del SMA-TAP en capas*

10.3. DESCRIPCIÓN DEL FLUJO EN CONEXIONES PUNTO-PUNTO

En esta sección vamos a presentar los flujos de datos y de control entre los diferentes elementos que constituyen la arquitectura TAP, en un intento por aclarar el funcionamiento de los conmutadores AcTMs que constituyen la VPN formada por nodos activos y multiagente. Para ello vamos a usar la *Figura 10.8* donde hemos etiquetado con un número cada una de las etapas que consideramos de mayor interés.

Antes de comenzar el análisis pormenorizado de cada uno de los apartados destacamos que en la *Figura 10.8* se han representado también detalladamente todos los bloques que componen la arquitectura de los nodos activos, así como cada uno de los agentes software que controlan, tanto el flujo de datos, como el control. Presentamos por esto los agentes diferenciados entre los que son programables y los que no lo son. Del mismo modo, se representa con líneas de trazo diferente el tráfico de datos, el de control y el provocado por las solicitudes de retransmisión entre conmutadores adyacentes. Aparecen también representadas las interacciones entre los agentes CoSA-WFQA y entre WFQA-RCA. Queremos destacar también la posibilidad de comunicación entre agentes RCA de conmutadores adyacentes que se realiza a través de las células BRM indicadas para las retransmisiones. La *Figura 10.8* no representa la comunicación entre agentes CoSA y CCA entre conmutadores AcTMs consecutivos. Seguidamente iremos describiendo cada una de las etapas de forma secuencial, tal como sería el flujo normal de datos dentro de los conmutadores activos.

① El agente programable CoSA es el encargado de la función de establecimiento de conexión en la red. Por esto permite identificar cada una de las fuentes de tráfico que se van a poner en marcha, y por tanto, otra de sus funciones es la de caracterizar los parámetros de tráfico de cada una de las fuentes. Una vez establecida la conexión, asignadas las tablas de routing e identificados los parámetros de tráfico, mantiene la comunicación con cada uno de los agentes CoSA del resto de los conmutadores activos presentes en la red y que están implicados en las conexiones negociadas.

② Antes de comenzar la comunicación es necesario actuar sobre el agente programable CCA para indicar los mecanismos de control de congestión que desean aplicarse a los flujos de datos. Es decir, elegir el umbral de EPDR, decidir si se desea usar PPD u otro algoritmo de control de congestión sobre el buffer etc. Las características definidas por CCA sobre el buffer afectarán a todo el tráfico de datos; sin embargo, al tratarse de un agente programable permitirá cambiar en tiempo de ejecución el valor del umbral cuando se están detectando excesivas congestiones.

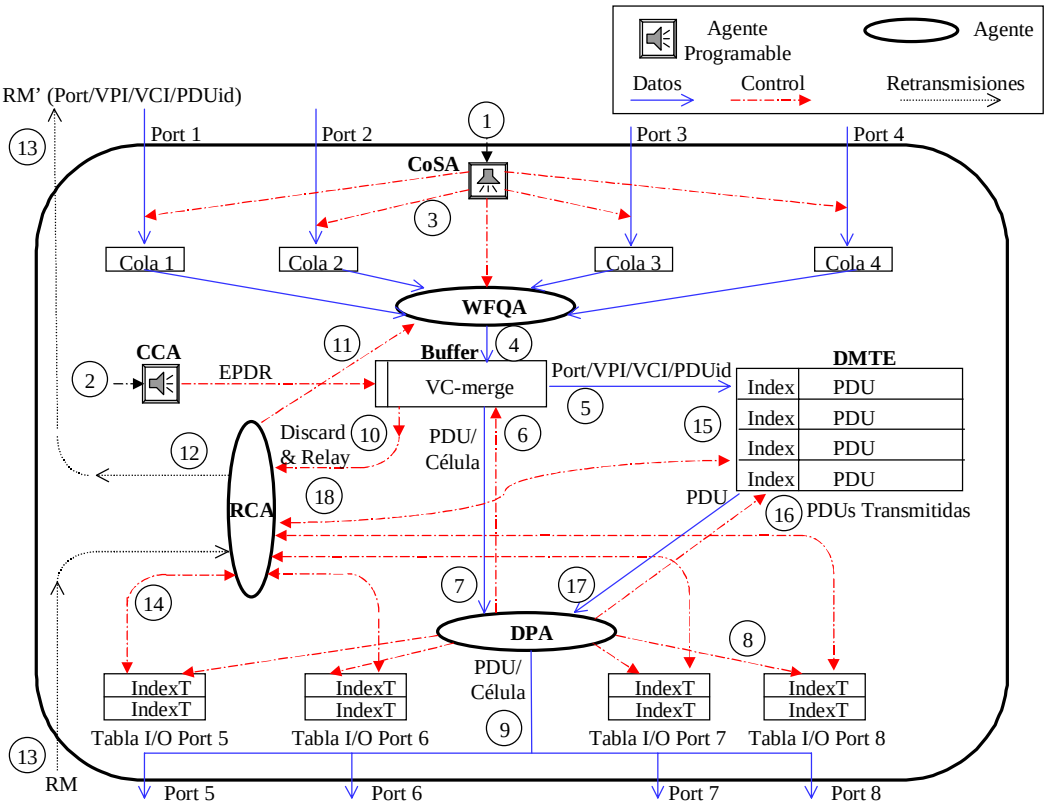

Figura 10.8. *Flujo de datos y control de un conmutador AcTMs*

③ Una vez elegidas las características del tráfico y del buffer comienza la transferencia desde las fuentes que generan PDU en el caso de las fuentes privilegiadas y células ATM nativas si se trata de fuentes de background. El tráfico llega a cada una de las colas correspondientes de los puertos de entrada de forma que, como sabemos, cada VPI/VCI tendrá su propia cola de entrada en el conmutador. Tal como hemos descrito anteriormente tendremos separado el flujo en colas diferentes para cada una de las fuentes. En este punto es donde se establece la comunicación del agente CoSA con el agente WFQA que entra en acción en el siguiente punto.

④ El agente WFQA se encarga de servir cada una de las cabeceras de las colas según lo que hemos comentado en secciones anteriores. De este modo el flujo de datos irá llegando al buffer del conmutador en forma de PDU y/o de células, donde van a ser atendidas según las características especificadas en el punto ② por el agente programable CCA.

⑤ Una vez que las datos están en el buffer, se realiza la función de copia de las PDU en la memoria DMTE, que es indexada con los datos de la PDU (*VPI/VCI/PDUid/Port*). Para ello se usa la función *copy(index,PDU)*, pero antes de depositar la PDU en la memoria son comprobadas las PDU ya existentes de cada conexión para garantizar el grado de GoS ya comentado. De esta forma, si ya se tienen en memoria el número de PDU máximo, la que acaba de llegar elimina la de PDUid más pequeño. El índice empleado en la entrada de la PDU será posteriormente empleada en los accesos para lectura en el caso de las solicitudes de retransmisión.

⑥ Una vez realizada la copia de la PDU entrante en la DMTE el buffer espera desde el agente DPA una petición de envío. Para esto el agente DPA dispone de la función *get (siguiente)*. Así, el *dispacher* se encarga de comprobar si existen PDU o células pendientes de procesar en el buffer, de modo que cuando su nivel de actividad lo permite, solicita que le sea enviada la siguiente unidad de transferencia a enviar a los puertos de salida.

⑦ Cuando el buffer recibe una orden de *get(siguiente)* desde el agente DPA, se activa la función *send(siguiente)* que es la responsable de enviar al *dispacher* cada una de las PDU que han sido previamente copiadas en la memoria DMTE o de las células independientes que no se copian en memoria. La función *get* debe sincronizarse adecuadamente con las funciones *copy* y *send* asociadas al buffer. Cuando *get* obtiene respuesta positiva, *copy(index,PDU)* debe haber terminado de realizar la copia de la PDU solicitada en la DMTE antes de enviarla al agente DPA. Una vez enviada cada PDU o célula, éstas son eliminadas del buffer liberando espacio para el tráfico entrante que pueda estar a la espera en las colas de entrada. Tenemos que destacar que las funciones *get* y *send* son las que se encargan de garantizar la operación atómica de envío desde el buffer que permite que puedan procesarse todas las células de una PDU de forma completa evitándose así el *interleaving*.

⑧ En este punto el agente habrá recibido la siguiente PDU/célula desde el buffer, de forma que deberá encargarse de enviar cada unidad de transferencia a su correspondiente puerto de salida. Pero antes de realizar el envío deberá ocuparse de la operación de asignación de VPI/VCI a las células que le hayan llegado. Cada vez que el *dispacher* recibe una PDU destinada a un *port* de salida, se encarga de actualizar su correspondiente tabla de E/S para relacionar los puertos de E/S y tener el índice de acceso a la DMTE en las retransmisiones. Por tanto, el agente DPA debe acceder a la correspondiente tabla de E/S del puerto de salida para registrar el índice de la PDU que se va a enviar a través de ese puerto. Este índice es el mismo que se ha utilizado en la DMTE, sólo que se le añade también el puerto de salida de cada PDU garantizada para evitar la posible repetición entre los VPI/VCI de conexiones diferentes.

⑨ Actualizadas todas las estructuras, el *dispacher* envía cada PDU a su correspondiente *port* de salida hasta el siguiente conmutador en dirección al destino que puede ser un nodo activo o no. Los datos enviados entre conmutadores son células ATM nativas como si de una troncal ATM se tratase.

⑩ En este punto volvemos de nuevo al buffer del conmutador para explicar la forma de proceder en el caso que se produzca una congestión del mismo. Hasta ahora hemos visto el flujo de datos en la situación en que las PDU tienen cabida en el buffer, pero cuando no es así entra en acción el algoritmo EPDR que se encarga de descartar las PDU que no tienen cabida completa en el buffer al superar el valor del umbral. Cuando esto se produce actúa el agente RCA que se encarga de evitar la fragmentación de las PDU que superan el valor del umbral del buffer. Al RCA se pasan los datos de la PDU mediante el valor del índice de la misma.

⑪ La primera función que debe realizar el agente RCA al recibir una notificación de congestión desde el buffer es la de comunicar al agente WFQA la solicitud que va a realizar de retransmisión al conmutador anterior. De este modo el agente WFQA tendrá conocimiento de la llegada inminente de una PDU retransmitida que deberá priorizar sobre el resto de PDU que estén en las colas.

⑫ El siguiente paso realizado por el agente RCA es el de preparar la célula BRM que debe enviar al conmutador anterior, en la que introduce el índice de la PDU que ha sido descartada y que hay que retransmitir desde conmutadores anteriores. Antes de preparar la PDU RCA consulta si el enlace afectado dispone de suficiente tiempo de OFF para poder atender la petición de retransmisión. Realizados estos pasos la BRM es enviada por el VPI/VCI reservado al efecto.

⑬ Como puede comprobarse en la *Figura 10.8*, este paso está etiquetado en dos puntos diferentes, aunque en realidad hace referencia al mismo proceso. Con esto queremos indicar que la solicitud de retransmisión puede realizarse desde dos puntos diferentes en cada conmutador activo. Es decir, en cada AcTMs puede generarse una BRM en el agente RCA, pero también puede llegarle una BRM desde el conmutador que le sigue que habrá, por tanto, experimentado una congestión. Esta etiqueta identifica entonces el VPI/VCI reservado para las células RM, en las que se indica el índice de la PDU según se ha explicado ya en capítulos anteriores. Destacamos que la etiqueta de la parte superior de la Figura acabará llegando al conmutador anterior que atenderá la RM desde el agente RCA, tal como lo hace el que está representado en la Figura. En el caso que el conmutador anterior no sea un AcTMs, y no soporte TAP, la célula RM será reenviada hasta el siguiente conmutador en dirección al emisor, y así sucesivamente hasta encontrar un conmutador activo que sea capaz de atender la solicitud de retransmisión, o hasta llegar a la fuente de tráfico.

⑭ Cuando al RCA llega la célula RM de retransmisión, el agente accede al campo que contiene el índice de la misma. Mediante este índice se accede a la correspondiente tabla de E/S del conmutador para poder obtener la relación de puertos de E/S y así tener el índice real de acceso a la memoria DMTE. Una vez conseguido el índice de la tabla de E/S es pasado al agente RCA.

⑮ Cuando conocemos el índice de acceso a la memoria DMTE se intenta localizar en ésta la PDU que el conmutador siguiente ha perdido por congestión. Una vez que se ha accedido a la memoria mediante la función de *hash*, pueden ocurrir dos cosas: que la PDU esté en memoria aún, o que ya no se encuentre en la misma, porque hayan llegado otras PDU con posterioridad que hayan ocupado el espacio de memoria de las PDU más antiguas de esa conexión concreta, dando lugar a lo que llamamos fallo de memoria DMTE.

⑯ Si la PDU solicitada está aún en la memoria, el protocolo TAP que se ejecuta dentro del conmutador y controla el buffer y la DMTE, lo notifica al agente DPA que solicita la PDU a la DMTE para retransmitirla. En realidad, este paso es parecido al ya descrito en la etiqueta ⑥, porque se dispone de una operación *get* similar para solicitar el envío de una PDU en este caso desde la DMTE en lugar de hacerlo desde el buffer.

⑰ Una vez solicitada la PDU desde el DPA, es la memoria DMTE la que se encarga de servirla al *dispacher* del mismo modo que se hacía en la etiqueta ⑦ desde el buffer. Por tanto, en este caso también se recurre a la atomicidad para evitar el *interleaving*, tal como ocurría en el paso ya descrito. Una vez que la PDU retransmitida llega al DPA se repetirán los pasos ⑧ y ⑨ descritos anteriormente, para reenviar la PDU hasta el conmutador siguiente que se congestionó en la fase anterior.

⑱ Este paso de la secuencia se da cuando en la DMTE no existe la PDU solicitada. RCA de este modo conoce que la PDU no ha sido localizada por lo que debe solicitar la retransmisión hacia el conmutador previo. Esta etapa coincide realmente con la indicada en la etiqueta ⑩ y, por tanto, se llega al agente RCA con una PDU no localizada (en lugar de congestionada en el conmutador actual), pero que RCA debe resolver de la misma forma que si la petición viniese del buffer. El mecanismo de solicitud de retransmisiones puede continuar en sentido al emisor hasta localizar la PDU en un AcTMs, teniendo en cuenta que los conmutadores no activos sólo se encargan de reenviar la BRM en sentido contrario al flujo de datos hacia el conmutador previo.

Antes de concluir hemos de destacar que acabamos de describir un escenario de comunicación punto-punto, y que las extensiones necesarias para las transferencias punto-multipunto, multipunto-punto y multipunto-multipunto sólo requieren de la incorporación en los nodos emisores y en el resto de los nodos

activos de la red, de las rutas o secuencias de VPI/VCI de las múltiples fuentes de tráfico y receptores de las conexiones multipunto. En general, en ATM las investigaciones relacionadas con las comunicaciones multipunto apuntan en la dirección de la unión de *n* comunicaciones punto-punto.

10.5. CONCLUSIONES

En este capítulo se ha presentado el detalle de cada uno de los bloques que componen la arquitectura de los conmutadores activos AcTMs. La memoria DMTE desempeña un papel fundamental para conseguir aportar la GoS a las fuentes privilegiadas, pero otros componentes como las colas de entrada permiten la separación de los flujos en colas separadas a la vez que permiten establecer pesos a cada uno de los VCI con la intención de poder aportar un tratamiento específico, pero a la vez justo a cada una de las fuentes. El buffer del conmutador también desempeña una función básica en la arquitectura ya que con una adecuada gestión del mismo se consiguen amortiguar muchos de los efectos negativos descritos en el *Capítulo 9*. Las Tablas de E/S aportan los índices de acceso a la memoria DMTE en las retransmisiones y nos dan la relación entre los puertos de entrada y de salida en cada una de las conexiones privilegiadas. Toda esta equipación hardware es gestionada por los mecanismos software que aportan los cinco agentes que conforman el SMA-TAP cuyo funcionamiento también ha sido descrito en este capítulo. Se han justificado determinadas decisiones de diseño como el tamaño de la DMTE, el de las colas de entrada y el del buffer, con la intención de demostrar que su coste está justificado por la GoS aportada y por la optimización del goodput que aportan a la red. Una vez analizados los bloques separadamente se han estudiados los flujos de datos y del control en conexiones punto-punto para comprender su funcionamiento.

REFERENCIAS

[1] J. M. Pitss, “Introduction to ATM. Design and Performance”, *Ed. Wiley*, (1997).

[2] M. de Prycker, “Asynchronous Transfer Mode. Solution for Broadband ISDN (3ª Ed.),” *Ed. Prentice Hall*, (1995).

[3] Handël, R., Huber, M. N., Schoröder, S., “ATM Networks Concepts, Protocols. Applications (2ª Ed.),” *Addison-Wesley*, (1995).

CAPÍTULO 11

PROTOCOLO TAP (TRUSTED AND ACTIVE PROTOCOL)

11.1. INTRODUCCIÓN

En el capitulo anterior hemos descrito cada uno de los componentes de la arquitectura TAP, poniendo especial atención en los bloques hardware de los conmutadores activos AcTMs. Hemos comentado ya que cada uno de esos bloques hardware son controlados por diversos algoritmos que, en la mayor parte de los casos, se corresponden con la implementación de los agentes que componen la arquitectura. Por tanto, la arquitectura hardware propuesta está equipada con una importante faceta software que constituye el protocolo TAP (*Trusted and Active Protocol*) es implementado sobre la propia arquitectura hardware.

En el punto 3.2 del *Capítulo 3* hemos justificado el calificativo *trusted* de nuestro protocolo, en comparación con el concepto *reliable*. Como sabemos, TAP busca la aportación de la garantía de servicio a las conexiones que lo requieran y que puedan verse afectadas por las impredecibles congestiones de los conmutadores tradicionales. Nuestra aportación *trusted* es, por tanto, un complemento para los *reliable protocols* que aportan la fiabilidad a las fuentes que pueden experimentar errores en sus transferencias.

Por otro lado, la característica *active* de TAP ha sido argumentada en el apartado 6.11 del *Capítulo 6*, donde hemos introducido las ventajas de aportar a las redes de comunicaciones componentes software en forma de agentes que las conviertan en programables. Los agentes software que hemos incorporado a la arquitectura TAP convierten a los conmutadores que los soportan en elementos activos en cuanto al procesamiento de los flujos de datos y a las propias labores de control de la red. La incorporación de los agentes software en nuestra propuesta nos permite, por un lado identificar a los conmutadores que los incorporan como conmutadores ATM activos (que denominamos AcTMs) y, por otro, definir al protocolo distribuido sobre la VPN constituida por los AcTMs también como activo.

Además, el *Capítulo 7* nos ha permitido extractar las ventajas de los sistemas y las arquitecturas distribuidas y que hemos incorporado a la visión que pretendemos dar de la red privada virtual sobre la que se ejecuta TAP, que se convierte así en un protocolo distribuido a lo largo de toda la red. Esta visión distribuida permite evitar las dependencias de nodos centralizadores, de forma que TAP funciona independientemente de las características de los nodos que componen la VPN. Es decir, pueden existir nodos en la red que no soporten TAP. Además, no existe ningún nodo que gestione la información completa sobre el estado de la red, de modo que cada conmutador puede tomar decisiones basándose sólo en su información local. Todo esto acaba beneficiando a la integridad, eficacia y robustez del protocolo.

A la vista de todo lo anterior puede entenderse el protocolo TAP como un conjunto de algoritmos distribuidos a lo largo de toda la red, algunos de los cuales aportan la característica activa a los conmutadores, mientras otros se responsabilizan de ofrecer la garantía de servicio a aquellas conexiones que deciden usarlos. Según esto, dispondremos de un algoritmo principal que es ejecutado extremo-extremo entre los terminales de la comunicación, y del que dependen el resto de algoritmos que son implementados en los conmutadores AcTMs de la red. A todo el conjunto de algoritmos lo denominamos TAP y hemos de destacar que existe una relación muy estrecha entre ellos y el subsistema que constituye el SMA-TAP.

En este capítulo, por tanto, se van a describir las consideraciones más importantes acerca de cada uno de los algoritmos que componen TAP, con la intención de conocer sus detalles de implementación y su funcionamiento. Destacamos que los algoritmos presentados son sólo una descripción de las funciones a realizar en el simulador de TAP, más que de la propia implementación hardware de los AcTMs que no es objetivo de esta tesis. Los algoritmos intentan dar una idea de la forma en que se realizarían por software las funciones más importantes; sin embargo, la arquitectura hardware final permitirá realizar las funciones de transferencia de forma más eficiente. En este capítulo demostraremos además la base de funcionamiento de nuestra idea de aprovechar los estados de inactividad de las fuentes de tráfico para atender las solicitudes de retransmisión, de forma que no se afecte al rendimiento de la red. Este es uno de los fundamentos de la filosofía de funcionamiento de TAP y para estudiarlo nos apoyaremos en la teoría de fuentes ON/OFF, que serán también empleadas en el prototipo de TAP que hemos desarrollado y que será explicado en el *Capítulo 12*. Estudiaremos también en este capítulo las implicaciones sobre la señalización de la red.

11.2. PROTOCOLO TAP EN LOS NODOS TERMINALES Y ACTMS

En primer lugar vamos a describir el algoritmo TAP desde el punto de vista de los extremos terminales de la comunicación. Para ello nos vamos a basar en un escenario con un canal unidireccional y punto-a-punto. Destacamos que los algoritmos presentados se centran principalmente en los aspectos de mayor interés para entender el comportamiento de TAP. Por esto hemos obviado diversos aspectos estándares de la propia tecnología ATM (control de errores, segmentación y reensamblado, primitivas, etc.) para no distraer la atención del objetivo principal de este capítulo. Con la intención de no hacer esta descripción demasiado extensa presentamos también un diseño estructurado y en sus primeros niveles de abstracción, por lo que no entramos a describir los detalles de implementación de las funciones empleadas.

El primer aspecto que debemos comentar es la implementación del protocolo sobre el terminal emisor, en el que se soportan todas las capas que fueron presentadas en la *Figura 10.1*, donde puede observarse que TAP está situado en el modelo ATM por debajo de protocolos de capas superiores, y por encima de nuestra propuesta EAAL-5. De este modo, el terminal emisor debe encargarse de los primeros aspectos de la comunicación y, tal como muestra la *Figura 11.1*, podemos comprobar cómo el algoritmo TAP negocia los parámetros de la comunicación y, además, se encarga de iniciar todos los demás algoritmos del conjunto que constituyen el protocolo completo. Por esto inicia y pasa los parámetros a cada uno de los agentes que están soportados en los nodos AcTMs de la VPN.

Una vez iniciada la conexión extremo-extremo entre los dos nodos terminales y la secuencia de PDU garantizadas, podemos observan en la *Figura 11.1* cómo el nodo emisor recibe los paquetes del tráfico desde capas superiores que serán después debidamente procesados por EAAL-5. Cuando concluye el tiempo de la transferencia, o se decide liberar la conexión por alguno de los extremos, termina también el algoritmo TAP en el nodo emisor.

```
Negociacion_Conexion(VPI,VCI,Ton,Toff,PCR,tiempoSimul,gradoGoS,ToS,algoritmo,umbral)
Inic_CoSA(VPI,VCI);              /* Inicia el agente de Clase de Servicio */
Inic_CCA(algoritmo,umbral);      /* Establece el algoritmo y umbral a aplicar sobre el buffer */
Inic_DMTE(VPI,VCI,gradoGoS);     /* Determina Nº PDU almacenadas en DMTE por conexión */
Inic_RCA(evento);                /* Inicia el agente RCA */
Inic_WFQA(VPI,VCI,PCR);          /* Inicia el agente WFQA y establece peso de cola para VCI*/
Inic_DPA(evento);                /* Inicia el agente DPA */
Inic(tiempoSimul);               /* Activa el reloj de duración de la simulación */
PDUid:=0;

Mientras tiempoSimul > 0 o Liberacion_Conexion(VPI,VCI) ="N"
      paq_TAP:=Desde_Protocolo_Superior(paquete);  /* Toma paquete de capa superior */
      Hasta_EAAL-5(VPI,VCI,paq_TAP);               /* Pasa el paquete a la capa EAAL-5 */
FMientras;
Liberacion_Conexion(VPI,VCI);    /* Concluida la transferencia se libera la conexión */
```

Figura 11.1. *Algoritmo TAP en el terminal emisor*

La *Figura 11.2* presenta el algoritmo TAP que se ejecuta sobre el terminal receptor por lo que, en este caso, el flujo de datos sigue el sentido contrario al del nodo emisor. Es decir, en el receptor se reciben las PDU desde la capa EAAL-5 para ser transferidas hasta los protocolos de capas superiores. La comunicación concluye de forma similar a lo visto en el caso del nodo emisor.

```
Mientras tiempoSimul > 0 o Liberacion_Conexion(VPI,VCI) = "N"
      paq_TAP:=Desde_EAAL-5(VPI,VCI,PDU);   /* Recibe PDU desde capa EAAL-5 */
      Hasta_Protocolo_Superior(paq_TAP);    /* Envía la PDU al protocolo superior */
FMientras
Liberacion_Conexion(VPI/VCI);               /* Concluida la transferencia se libera la conexión */
```

***Figura 11.2**. Algoritmo TAP en el terminal receptor*

11.2.1. EAAL-5 (EXTENDED ATM ADAPTATION LAYER 5)

El siguiente algoritmo que vamos a comentar es precisamente el de la capa EAAL-5 del nodo emisor que está debajo del algoritmo TAP que se comentó en la *Figura 11.1*. Este algoritmo presentado en la *Figura 11.3* se encarga, principalmente, de generar en el terminal emisor las unidades de PDU de las conexiones privilegiadas. De este modo, se asigna a cada PDU su identificador correspondiente que, como ya sabemos, es la base del mecanismo de recuperación de congestiones. Una vez generadas las PDU, éstas son segmentadas en células independientes y transferidas a la capa ATM estándar que se encargará de ponerlas en la red a través de la capa Física del emisor. Puede observarse cómo cuando se agota la secuencia de 65.536 PDU generadas, ésta es reiniciada nuevamente.

```
Mientras No_Liberacion_Conexion(VPI,VCI);
      paq:=Desde_TAP(VPI,VCI,paq_TAP);          /* Recibe un paquete desde la capa TAP */
      Mientras NoFin(paq)
            PDU:=Generar_PDU(VPI,VCI,paq,PDUid); /* Genera PDU y le asigna PDUid */
            PDUid:=PDUid+1;
            Mientras NoFin(PDU)
                  celula:=SAR(VPI,VCI,PDU);     /* Segmentación de la PDU  en células */
                  Hasta_ATM(celula);            /* Pasa cada célula a la capa ATM del emisor */
            FMientras;
            Si PDUid=65.536 entonces PDUid:=0   /* Inicia la secuencia de numeración de PDU */
      FMientras;
FMientras;
```

***Figura 11.3.** EAAL-5 en el terminal emisor*

A continuación nos encontramos con el algoritmo EAAL-5 ejecutado en el nodo receptor de la conexión. Como puede comprobarse en la *Figura 11.4*, en este caso el sentido del flujo es el contrario al de la *Figura 11.3*, ya que ahora se reciben las células desde la capa ATM, por lo que EAAL-5 se encarga de reensamblarlas en unidades de PDU que, una vez localizada su célula final (EOM), son transferidas hasta la capa en que se encuentra el algoritmo TAP. Como vimos antes, TAP se responsabilizará de hacerlas llegar a los protocolos superiores.

```
Mientras No_Liberacion_Conexion(VPI,VCI)
      Hasta EOM(VPI,VCI,cel)
            cel:=Desde_ATM(celula);         /* Recibe una célula desde la capa ATM */
            PDU:=SAR(VPI,VCI,cel);          /* Reensamblado de las PDU */
      FHasta;
      Si ToS="D" Hasta_TAP(VPI,VCI,PDU)     /* Pasa PDU al protocolo TAP */
      En otro Caso entonces
            Si Longitud <= ventana_ordenacion entonces Comenzar
                  Ordenar(VPI,VCI,PDUid,ListaPDUs);  /* Servicio ordenado */
                  Longitud:=Longitud(ListaPDUs)+1;
            FSi;
      FSi;
      Si Longitud = ventana_ordenacion entonces
            Mientras Longitud > 0
                  PDU:=Cabecera(ListaPDUs);
                  Hasta_TAP(VPI,VCI,PDU);       /* Pasa PDUs ordenadas a TAP */
                  Longitud:=Longitud-1;
            FMientras;
      FSi;
FMientras;
```

***Figura 11.4.** EAAL-5 en el terminal receptor*

En este caso EAAL-5 se debe encargar también de, en función del tipo de servicio (ToS) elegido en el establecimiento de la conexión por el terminal emisor, hacer la ordenación de las PDU que se reciben. Como ya sabemos por capítulos anteriores, el mecanismo de retransmisiones puede acabar generando desorden en las PDU, por lo que el protocolo dispone de dos tipos de servicio diferentes: uno ordenado y otro secuencial. Mientras el primero se encarga de pasar a la capa TAP las PDU ordenadas y detectando fallos de secuencia o pérdidas; el segundo transfiere a TAP las PDU nada más ensamblarlas, delegando la ordenación y detección de pérdidas a los protocolos superiores. Mientras el primer servicio es más elaborado y costoso en cómputo, el segundo es menos cuidadoso pero más rápido. Recordamos antes de concluir que el algoritmo EAAL-5 es sólo implementado en los equipos terminales de la comunicación, y no en los nodos AcTMs que sólo consultan los identificadores de PDU que procesan. Esto es así para evitar introducir retardos en la red, y también para mantener las características estándares de ATM, donde los conmutadores sólo disponen de las capas Física y ATM.

11.2.2. ALGORITMO TAP SOBRE LOS ACTMS

Los nodos terminales de la comunicación soportan el protocolo TAP descrito en las figuras anteriores e, igualmente, éstos incluyen la extensión EAAL-5 que acabamos de comentar. Sin embargo, los nodos activos de la red no se equipan con EAAL-5 para no afectar al rendimiento de la red y, del mismo modo, tampoco soportan las características del protocolo TAP que hemos descrito en el caso de los nodos terminales emisor y receptor. No obstante, y dado que ya sabemos que TAP es un protocolo distribuido en toda la VPN constituida por los nodos AcTMs que la forman, queremos comentar algunos de los aspectos de los que se encarga esta variante de TAP que se ejecuta sobre los conmutadores activos.

En cierta forma, este algoritmo que vamos a comentar actúa a modo de columna vertebral sobre la que se articulan el resto de algoritmos que funcionan en los AcTMs. Esto es así porque el algoritmo de la *Figura 11.5* se encarga de copiar las PDU fiables desde el buffer a la DMTE, y también de atender las peticiones del agente DPA, ya sean desde el buffer en las transmisiones, o desde la DMTE en el caso de retransmisiones. Para ello, este algoritmo necesitar cooperar con los agentes CCA y DPA.

```
Sincronizacion_CCA(VPI,VCI,PDUid);                  /* Sincronización con el agente WFQA local */
Sincronizacion_DPA(evento);                         /* Sincronización con otros agentes RCA de la VPN*/
Inic_DMTE(VPI,VCI,gradoGoS);                        /* Determina Nº PDU almacenadas en DMTE por conexión */
Mientras cierto
    Si Fincelula() entonces put(buffer,célula);     /* Notificación desde CCA de fin de célula en buffer */
    Si FinPDU(VPI,VCI,PDUid) entonces Comenzar      /* Notificación desde CCA de fin de PDU en buffer */
        h:=hash(VPI,VCI,PDUid);                     /* Función de hash de acceso a la DMTE */
        g:= Inic_DMTE(VPI,VCI,gradoGoS);            /* Obtenemos el grado de GoS de la conexión */
        Si get(buffer,PDU) entonces Comenzar        /* Si solicitud de envío desde el agente DPA */
            Si NumPDUVCI(DMTE,h) >= g entonces
                                borrar(DMTE,h);     /* Elimina de la DMTE la PDU de id más pequeño */
            copiar(buffer,VPI,VCI,PDUid,DMTE,h);    /*Copia la PDU del buffer a la DMTE */
            put(buffer,PDU);                        /* Envía la PDU al DPA */
            borrar(buffer,PDU);                     /* Se elimina del buffer la PDU que se acaba de transferir */
        FSi;
        Si get(DMTE,PDU) entonces put(buffer,PDU);  /* Solicitud retransmisión de PDU desde la DMTE al DPA */
    FSi;
FMientras;
```

Figura 11.5. *Algoritmo TAP en los nodos AcTMs*

El bucle del algoritmo recibe desde CCA la notificación de fin de PDU cuando acaba de procesar una célula EOM en el buffer. Cuando esto ocurre, se calcula la función de *hash* para los accesos a la DMTE para cada PDU. Antes de enviar una PDU a la DMTE es necesario conocer el grado de GoS que se especificó para la conexión, ya que hay que controlar cuántas PDU de cada conexión se tienen ya almacenadas en la memoria. Para entrar una nueva PDU de una conexión que ya tiene el máximo número de PDU permitido (expresado por GoS) es necesario borrar previamente la PDU más antigua (*aging* sobre el PDUid) para liberar espacio de memoria. Una vez que se dispone de espacio, el algoritmo se encarga de realizar la copia de la PDU desde el buffer a la DMTE, para después enviar esa misma PDU hasta el agente DPA (que la habrá solicitado previamente con la función *get*). Después de enviar la PDU ésta es borrada para liberar el espacio ocupado en el buffer. Destacamos los accesos constantes *O(1)* a la tabla DMTE, gracias a la función de *hash*, que garantiza este coste mientras no aparezcan colisiones en los accesos a la tabla.

En el caso que el algoritmo reciba una petición de retransmisión desde DPA (mediante la operación *get*), entonces la PDU solicitada será enviada a este agente, tal como muestra la *Figura 11.5*. Este es el escenario en que una solicitud de retransmisión desde el conmutador siguiente al que está ejecutando el algoritmo, ha tenido éxito en la DMTE local, y donde ha sido localizada la PDU solicitada para su retransmisión.

11.3. AGENTE CoSA (CLASS OF SERVICE AGENT)

Como ya se ha explicado, este agente se encarga de atender diversos aspectos relativos a la clase de servicio que requieren de la red las fuentes de tráfico. La *Figura 11.6* formaliza las labores más destacables realizadas por el agente, así como las relaciones con el resto de agentes del SMA. El agente es inicializado por el terminal emisor de la comunicación desde donde se definen los parámetros de tráfico de cada una de las fuentes. Estos parámetros van a determinar, en cierto modo, la clase de servicio que desea cada fuente, entre los que se encuentra, por ejemplo, el grado de garantía de servicio que se requiere. De este modo, podemos decir que CoSA actúa como agente programable ya que, en función de sus características y acciones, se podrá lograr un mayor o menor grado de GoS, que es uno de los objetivos primordiales de TAP.

Además, tal como podemos comprobar en el algoritmo, este agente desempeña importantes funciones de comunicación y sincronización con el resto de agentes CoSA de la VPN en las labores de señalización y para el establecimiento de la conexión. Como ya hemos descrito en el *Capítulo 10*, el trabajo cooperativo de este agente con el agente WFQA es también imprescindible para lograr nuestros objetivos, por lo que en ambos se establecen las operaciones de sincronización, tanto para conseguir la GoS de las fuentes privilegiadas, como la justicia de todas las fuentes de tráfico en general, sean o no privilegiadas.

Otro de los aspectos destacables del algoritmo es el que este agente se encarga también de inicializar dos de las estructuras de datos más importantes de la arquitectura TAP, como son la memoria DMTE y las Tablas de E/S. Así, mediante la función *Inic_Grado_GoS_DMTE*, se realiza la reserva de memoria oportuna para soportar el número de PDU que se indica en el parámetro GoS para la conexión VPI/VCI. Por otro lado, las labores de establecimiento de la conexión, y sus implicaciones con la señalización, nos permiten determinar las relaciones de las conexiones con sus respectivos puertos de E/S en los AcTMs. Relaciones que deben ser registradas en la Tablas de E/S de cada puerto para poder obtener los índices de acceso a las PDU de la DMTE en el caso de retransmisiones. Todos estos aspectos han sido también aclarados en el *Capítulo 10*, y la función *Inic_TablaE/S* es usada en este agente para inicializar esta estructura al inicio de cada conexión.

El cuerpo del bucle del algoritmo de la *Figura 11.6* se encarga de procesar todas las células que le llegan desde la capa ATM de los AcTMs. En el caso que las células que llegan pertenezcan a conexiones privilegiadas éstas son ensambladas en PDU y transferidas al agente WFQA para llegar a sus correspondientes colas de entrada. En el caso que la conexión no sea privilegiada se transfieren las células independientes, también a su correspondiente cola de entrada al conmutador.

```
Inic_CoSA(VPI,VCI);                    /* Sincronización para el establecimiento de la conexión */
Sincronizacion_Agentes_CoSA(VPI,VCI);  /* Sincronización entre todos los agentes CoSA de la VPN*/
Establecimiento_Conexion (VPI,VCI);    /* Asignación de los VPI/VCI extremo-extremo */
Sincronizacion_WFQA(VPI,VCI);          /* Sincronización con al agente WFQA local */
Inic_Grado_GoS_DMTE(VPI,VCI,GoS);      /* Determinación de la GoS en la DMTE local */
Inic_TablaE/S(VPI,VCI,InPort,OutPortPrev); /* Iniciación de la TablaE/S local */
Mientras Desde_ATM(celula)             /* Se procesan todas las células que lleguen desde la capa ATM */
   Si Privilegiado(VPI,VCI) = "N"  entonces Hasta_WFQA(VPI,VCI,celula,p);
   En otro caso Comenzar               /* Cuando se trata de conexión privilegiada se ensambla la PDU */
        PDU:=Ensamblar_PDU(VPI,VCI,celula);
        Si EOM(VPI,VCI,celula) entonces Hasta_WFQA(VPI,VCI,c,PDU);  /* Pasar la PDU al agente WFQA */
   FSi;
FMientras;
```

Figura 11.6. *Algoritmo del Agente de CoS*

11.4. AGENTE WFQA (WEIGHTED FAIR QUEUEING AGENT)

Aunque el objetivo principal de TAP es el de ofrecer GoS a las fuentes privilegiadas, también ofrece la ventaja añadida de la atención justa del tráfico generado por todas las fuentes, sean privilegiadas o no. Nos encontramos con la posibilidad que al intentar garantizar las transferencias con GoS, podría darse la posibilidad que las fuentes no privilegiadas acabasen siendo relegadas. Para controlar esta situación

proponemos el algoritmo QPWFA que permite asignar pesos a las colas de entrada de los AcTMs en función de los valores de PCR de las fuentes. Para evitar la inanición de fuentes con PCR reducido, con respecto a las que lo tengan más elevado, incluimos el control sobre la longitud de las colas de entrada antes de procesar el tráfico hasta el buffer del conmutador. El funcionamiento del algoritmo QPWFQ ha sido ya explicado en el apartado 4.4 del *Capítulo 4*, y los flujos en las colas de entrada que son gestionadas por QPWFQ pueden ser consultados en el punto 10.2.1 del *Capítulo 10*. Ahora vamos a complementar todos estos aspectos explicando el funcionamiento del agente WFQA (mostrado en la *Figura 11.7*) que, entre otras funciones importantes, implementa el algoritmo QPWFQ.

Como podemos comprobar en la *Figura 11.7*, el agente WFQA es también inicializado desde el terminal emisor de la comunicación, donde ya se indica el PCR de la fuente de tráfico, lo que acabará reflejándose como el peso que cada fuente tendrá asignado por el agente en su correspondiente *per-VC* cola de entrada.

```
Inic_WFQA(VPI,VCI,PCR);                    /* Sincronización para iniciar el agente de gestión justa de colas de entrada*/
Sincronizacion_CoSA(VPI,VCI);              /* Sincronización con el agente CoSA local */
Sincronizacion_RCA(VPI,VCI,PDUid);         /* Sincronización con el agente RCA local para atender retransmisiones*/
p_i:=Inic_WFQA(VPI,VCI,PCR);               /* Se asigna a la cola PerVC como peso el valor del PCR de la fuente i */
l_i:=0;                                    /* Longitudes de las colas de espera en cada momento */
Retransmision_RCA(VPI,VCI,PDUid):=falso;   /* No hay retransmisiones al inicio */

Mientras Desde_CoSA(VPI,VCI,celula,PDU)    /* Se procesan las PDU o células llegadas desde CoSA */
    i:=VCI;                                /* Asignación de la PDU o de la célula del VCI i a la cola i */
    Si Retransmision_RCA(VPI,VCI,PDUid) entonces comenzar
        encolar(ColaVPI/VCIi,c,PDUid);     /* Encolar PDU en cola VPI/VCIi */
        apuntar(ColaVPI/VCIi,PDUid);       /* Apunta PDU retransmitida como más prioritaria a la cabecera de la cola i */
    Sino encolar(ColaVPI/VCIi,celula,PDU); /* Introduce en cola de datos i la PDU o célula del VCI i */
    FSi;
    l_i:=l_i+1;                            /* Cuando llega una PDU o célula incrementa en 1 la longitud de su cola */
    Si (l_i<=p_i o Retransmision_RCA(VPI,VCI,PDUid)) entonces comenzar
      Caso
        Retransmision_RCA(VPI,VCI,PDUid):  encolar(turnos,VCIi); /* Encola en turnos VCI=i como más prioritario que cabecera */
        l_i<=p_i : encolar(turnos,VCIi):   encolar(turnos, VCIi); /* No se trata de una retransmisión */
      FCaso;
    FSi;
FMientras;

Mientras Novacias(colas_datos)             /* Si hay PDUs o células pendientes de transmitir o retransmitir en las colas de entrada */
    desencolar(turnos,i);                  /* Obtener el turno i de la cabecera de cola de turnos o del puntero a PDU retransmitida */
    Hasta_Buffer(ColaVPIVCIi,PDU,celula);  /* Envío de la PDU o célula de la cola i hasta el buffer */
    desencolar(ColaVPIVCIi,PDU,celula);    /* Eliminar PDU o célula de la cola i enviada al buffer */
    l_i:=l_i-1;                            /* Decrementa en uno la longitud de la cola de espera procesada */
    Si (l_i>=p_i) y (No_Retransmision_RCA(VPI,VCI,PDUid) y resto_de_colas_idle) entonces
        encolar(turnos,VCIi);              /* Reencola el turno VCI=i en la cola de turnos garantizando política work-conserving */
    FSi;
FMientras;
```

Figura 11.7. *Algoritmo del agente WFQA*

El algoritmo muestra también las funciones de sincronización del agente WFQA con el agente CoSA con los objetivos que ya hemos comentado en el apartado anterior. Además se establece también la necesaria sincronización con el agente RCA que es el encargado, como hemos explicado en el *Capítulo 10*, de notificar a WFQA su solicitud de retransmisión de una PDU al conmutador anterior.

Las líneas 4 y 5 del algoritmo tienen como misión la de inicializar los pesos y las longitudes de las colas del conmutador, ya que este es el mecanismo ideado para aportar a la vez la justicia y el tratamiento privilegiado de las células de las fuentes de tráfico.

En el primer bucle de la *Figura 11.7* se reciben las células o PDU procesadas previamente por el agente CoSA. En primer lugar se realiza la labor de asignación *per-VC* para que cada conexión VPI/VCI tenga su propia cola, para pasar después a analizar la posibilidad de que haya llegado una solicitud de retransmisión (que deberá haber sido antes notificada por el agente RCA). Si es así, se encola la PDU en su correspondiente cola de entrada y se actualiza el puntero de prioridades para darle a esta PDU un tratamiento especial con respecto al resto que puedan existir ya en esa cola. En el caso de no ser una retransmisión, se encola la célula o PDU en su correspondiente *per-VC* para, posteriormente, incrementar en una unidad la longitud de ésta. Seguidamente se realiza la inclusión del número de la cola de datos en la cola de turnos que es la que lleva el orden de atención de las cabeceras de las colas de datos. Podemos observar cómo la condición de encolado en la cola de turnos es que la longitud de cada cola sea menor que el peso (PCR) asignado a la fuente (lo que evita los comportamientos injustos), o bien que se trate de una retransmisión.

El segundo bucle del algoritmo se encarga de servir los datos al buffer del conmutador en función del orden establecido por la cola de turnos, en cuya cabecera está el número de cola cuya célula o PDU será la siguiente en ser enviada *Hasta_Buffer*. Una vez procesadas las cabeceras de las colas, éstas son desencoladas y se decrementa en una unidad la longitud de la cola de datos procesada. Llamamos la atención sobre el hecho que las dos últimas líneas de este segundo bucle aportan la característica *work-conserving* al tratamiento de las colas que, como puede comprobarse, permite atender un VPI/VCI concreto sin atender al resto de colas, y siempre que éstas estén en estado de inactividad y/o no llegue una retransmisión.

La característica *work-conserving* permite optimizar el rendimiento del sistema por no requerir de tiempos de espera fijos para atender secuencialmente las colas. Además, es importante recordar, como explicamos en el *Capítulo 4*, el coste constante *O(1)* de mantenimiento y de acceso del sistema de colas de entrada y también el mínimo nivel de complejidad (juego reducido de variables y simplicidad en las sentencias de flujo del algoritmo) que acaba redundando en la optimización del rendimiento de la red.

11.5. AGENTE CCA (CONTROL CONGESTION AGENT)

Hemos identificado ya (*Capítulos 5,7,9* y *10*) el buffer como uno de los puntos clave de los AcTMs donde localizar los problemas relativos a las congestiones que deseamos evitar. El buffer es, por tanto, la principal fuente de problemas en cuanto a la aparición de congestiones por ser el punto de competencia entre las conexiones entrantes a los conmutadores. En nuestro caso disponemos de un agente que ofrece la posibilidad de elegir la política de evitación de pérdidas y respuesta ante las congestiones.

Este agente tiene también múltiples funciones encomendadas, aunque la más importante de ellas es la implementación del algoritmo EPDR ya descrito en el apartado 5.7 del *Capítulo 5*, donde la *Figura 5.8* presenta nuestro algoritmo EPDR, propuesto como variante del conocido EPD. En este caso, la *Figura 11.8* muestra el algoritmo del agente CCA que, como puede observarse, incluye EPDR. Seguidamente comentamos los aspectos más destacables de este algoritmo.

```
Inic_CCA(algoritmo,umbral);                          /* Inicializa Agente CCA con Algoritmo y umbral del buffer */
Sincronización_Agentes_CCA(algoritmo,umbral);        /* Sincronización con resto agentes CCA de la VPN */

Si Algoritmo="EPDR"
  Mientras Hasta_Buffer(VPI,VCI,celula)              /* Mientras llegan células al buffer */
     Si ConexionNo_Privilegiada(VPI,VCI) y buffer(lleno)="N" entonces insertar(buffer,celula);
     Si no Comenzar
          Si celula(VPI,VCI) en Lista-descartes entonces Comenzar   /* Una célula con este VPI/VCI ya ha sido descartada */
               Si EOM(celula) entonces Comenzar                     /* Es la última célula de una PDU */
                    Si Longitud_Actual_Cola < Tamaño_Maximo_Buffer entonces
                              insertar(buffer,celula);              /* La célula EOM cabe en el buffer y se inserta */
                    En otro caso
                              tirar(celula);                        /* Célula EOM no cabe en el buffer y es tirada */
                    FSi;
                    RCA_Retransmitir(VPI,VCI,PDUid);              /* Solicitud de retransmisión al agente RCA */
                    eliminar (VPI,VCI,Lista-descartes);           /* Se borra el VPI/VCI de la lista-descartes */
               En otro caso
                    tirar(celula);                                /* Célula no EOM que no cabe en el buffer y es tirada */
               FSi;
          En otro caso
               Si Longitud_Actual_Cola < umbral entonces Comenzar
                    insertar(buffer,celula);                      /* La célula cabe en el buffer y se inserta */
                    Si EOM(celula) entonces FinPDU(VPI,VCI,PDUid);/* Notifica a TAP del AcTMs el fin de una PDU en buffer */
               En otro caso Si (primera_célula(PDU) o (buffer(lleno)="S"))
                              tirar(celula);                      /* Evitar fragmentación de la PDU */
                              insertar (VPI,VCI,Lista-descarte); /* Marcar VPI/VCI con una célula ya descartada */
                         FSi;
               En otro caso comenzar
                    insertar(buffer,celula);                      /* Tamaño de buffer sobrepasa el umbral pero se acepta */
                    Si EOM(celula) entonces FinPDU(VPI,VCI,PDUid); /* Notifica a TAP del AcTMs el fin de una PDU en buffer */
               FSi;
          FSi;
     FSi;
  FMientras;
FSi;
```

Figura 11.8. *Algoritmo del agente CCA y EPDR*

CCA es también inicializado por el emisor de las conexiones, donde el usuario o administrador de la red, en el proceso de establecimiento de la conexión, especificará el algoritmo que desea emplear para controlar las congestiones del buffer. Este aspecto es el que aporta la característica programable a CCA, ya que está diseñado para que se pueda optar por diversos mecanismos de control de congestión (*Capítulo 5*) y por algunos de los parámetros de funcionamiento de estos mecanismos. En nuestro caso empleamos EPDR, por lo que permitimos elegir el umbral, pero podría optarse por EPD, PPD, o cualquier otro que desee implementarse como soporte de TAP que, al tener un diseño modular, permitirá optar por uno u otro en función de las necesidades concretas.

Puede observarse en la *Figura 11.8* también, como CCA establece mecanismos de comunicación con otros agentes de la VPN que soporten TAP, de forma que se mantenga una misma política de gestión del buffer en todos los conmutadores.

En el caso que al buffer lleguen células provenientes de conexiones no privilegiadas éstas no reciben ningún tratamiento especial, siendo insertadas en el buffer si éste no está lleno. En cambio, si se reciben desde las colas de entrada varias PDU, comienza a aplicarse la política de evitación de pérdidas y/o fragmentación de PDU en el caso de congestión que, como sabemos, depende del tamaño del umbral que se haya definido en el algoritmo EPDR. Podemos comprobar en la *Figura 11.8* cómo, en el caso de aparecer la congestión, se invoca al agente RCA que se encargará de iniciar el mecanismo de recuperación, actualizándose después la lista de VPI/VCI que ya ha experimentado descartes de algunas de sus células.

11.6. AGENTE DPA (DISPACHER PDU AGENT)

El algoritmo presentado en la *Figura 11.9* aclara las explicaciones dadas en el *Capítulo 10* sobre el funcionamiento del agente encargado de despachar las PDU desde el buffer hasta las correspondientes salidas de los conmutadores. El agente despacha, tanto PDU con GoS, como células pertenecientes a conexiones no fiables. Para ello es iniciado mediante un evento en el proceso de establecimiento de comunicación. Podemos ver también la relación de DPA, tanto con las Tablas de E/S, como con el buffer y con la DMTE.

```
Inic_DPA(evento);                          /* Inicializa Agente DPA para recibir células y PDU */
Buffer, DMTE, TablasE/S                    /* Acceso a estructuras de datos */

Mientras cierto
    Si retransmision(DMTE,PDU) entonces Comenzar   /* Sincronización con TAP para retransmisión desde DMTE */
        PDUR:=get(DMTE,PDU);               /* Solicita a la DMTE la PDU a retransmitir */
        retransmitir(PDUR);                /* Retransmisión de PDU desde la DMTE local */
    Si no  Comenzar
        paq:=get(buffer,siguiente)         /* Solicita al buffer la siguiente unidad de transferencia */
        Si celula(paq)="S" entonces Comenzar   /* Cuando se trata de una conexión no privilegiada */
            paq.VPI:=VPIOut;
            paq.VCI:=VCIOut;
            Hasta_ATM(paq);                /* Conmutación al nuevo VPI/VCI y envío directo a capa ATM */
        Si no Comenzar                     /* Estamos ante una PDU transferida completa desde el buffer */
            celulaEOM:=EOM(VPI,VCI,paq);   /* Obtenemos primero  la célula EOM de la PDU */
            Generar(IndexT,celulaEOM);     /* Se genera el índice de acceso a la TablaE/S */
            inserta(TablaE/S,IndexT);      /* Inserción del nuevo índice en la TablaE/S) */
            celulaEOM.VPI:=VPIOut;
            celulaEOM.VCI:=VCIOut;
            Hasta_ATM(celulaEOM);          /* Transferencia de la célula EOM hasta siguiente conmutador */
            Mientras No_Fin(paq)
                celula:=cell(VPI,VCI,paq);     /* Obtención del resto de las células de la PDU */
                celula.VPI:=VPIOut;
                celula.VCI:=VCIOut;
                Hasta_ATM(celula);         /* Se envía cada célula a la capa ATM del AcTMs */
            FMientras;
        FSi;
    FSi;
FMientras;
```

Figura 11.9. *Algoritmo del agente DPA*

Dentro del bucle principal del algoritmo se atienden en primer lugar las situaciones de retransmisión que pueden provenir desde la DMTE y notificadas desde el algoritmo TAP soportado en los AcTMs. Cuando DPA recibe esta notificación solicita con *get* el envío de la PDU desde la DMTE y, una vez que la tiene, se encarga de transmitirla al correspondiente puerto de salida del conmutador. La función *get* (junto con *put* desde el buffer o la DMTE) son las que aportan la atomicidad en el tratamiento de las PDU fiables que son tratadas con el mecanismo *VC-Merge*.

Cuando no existen retransmisiones se atiende el flujo normal de datos, siempre previa solicitud al buffer de la siguiente unidad disponible que puede ser una PDU o una célula, y que el buffer se encarga de servir atómicamente con su operación *put*. Si se procesa una célula independiente se le asigna su correspondiente valor de VPI/VCI de salida y se envía hasta la capa ATM del conmutador que las procesa a la salida. Si la siguiente unidad a transferir desde el buffer es una PDU con GoS, se comienza procesando el final de ésta para disponer del VPI/VCI/PDUid y se aplica la posible asignación de nuevos VPI/VCI de salida para poder enviar los datos en forma de células hasta la capa ATM. Antes de pasar las células a la capa ATM se genera el índice y se inserta en la correspondiente Tabla de E/S según lo explicado en el *Capítulo 10*.

11.7. AGENTE RCA (RETRANSMISSION CONTROL AGENT)

Este agente juega un importante papel en el mecanismo de recuperación de PDU perdidas por congestión, ya que es el responsable de recibir los eventos cuando aparecen las congestiones. Estos eventos pueden provenir, o bien desde el buffer local del conmutador en que está ejecutándose RCA, o bien a través de las células BRM que provienen del conmutador siguiente a donde se encuentra el agente que recibe el evento.

Podemos observar en la *Figura 11.10* el algoritmo de RCA, que es iniciado desde el emisor. Este agente colabora, como hemos visto ya, con WFQA y con el resto de agentes RCA de otros conmutadores desde donde, o hasta donde envía las células BRM.

```
Inic_RCA(evento);                                  /* Inicializa Agente RCA para recibir células y PDU */
Sincronizacion_WFQA(VPI,VCI,PDUid);                /* Sincronización con el agente WFQA local */
Sincronizacion_RCA(VPI,VCI,PDUid);                 /* Sincronización con otros agentes RCA de la VPN */

Mientras cierto
    Si RCA_Retransmitir(VPI,VCI,PDUid) entonces comenzar  /* Cuando llega una solicitud de retransmisión desde EPDR */
        Si Toff_agregado(enlace) entonces Comenzar    /* Si suficiente Toff en el enlace */
            Retransmision_RCA(VPI,VCI,PDUid);         /* Notificación de la solicitud de retransmisión al WFQA */
            GenerarBRM(celdaRM,VPI,VCI,PDUid);        /* Se genera la celda BRM para la solicitud de retransmisión */
            TransferirBRM(celdaRM);                   /* Transferencia de la BRM al conmutador previo */
        FSi;
    FSi;
    En otro caso
        Si LlegaBRM(celdaBRM) entonces comenzar       /* RCA recibe una BRM del conmutador siguiente a él */
            Index:=Obtener_Indice(celdaBRM);          /* Se obtiene el índice de la BRM */
            IndexDMTE:=Acceso(TablaE/S,Index);        /* Acceso a la TablaE/S para obtener el índice de acceso a DMTE */
            Si Esta_PDU(DMTE,Index)="S" entonces retransmision(DMTE,PDU);   /*Notifica retransmisión exitosa al DPA */
            Si no Si Toff_agregado(enlace) entonces Comenzar   /* Fallo de DMTE pues ya no contiene la PDU solicitada */
                Retransmision_RCA(VPI,VCI,PDUid);     /* Solicitud de retransmisión al conmutador previo */
                GenerarBRM(celdaRM,VPI,VCI,PDUid);    /* Se genera la celda BRM para solicitud retransmisión */
                TransferirBRM(celdaRM);               /* Transferencia de la BRM al conmutador previo */
            FSi;
        FSi;
    FSi;
FMientras;
```

Figura 11.10. *Algoritmo del agente RCA*

Cuando RCA recibe una solicitud de retransmisión desde el buffer local al conmutador, la primera labor que realiza es evaluar el tiempo de inactividad de que se dispone en el enlace y, si es suficiente, comienza el proceso de recuperación, notificando en primer lugar la situación a WFQA, para pasar después a generar la BRM con los datos de la PDU a retransmitir. Una vez generada la célula RM, ésta es enviada al conmutador previo a través del VPI/VCI estándar reservado para las células RM.

En cambio, como puede observarse, cuando la solicitud de retransmisión no proviene del buffer local, sino que llega en una RM desde el conmutador siguiente que puede haber sufrido una congestión, lo primero que se hace es obtener el índice de acceso a la DMTE local para localizar la PDU cuyo PDUid viene indicado en la RM. Obtenido el índice de la célula RM, se accede a la correspondiente Tabla de E/S en la que se

encuentra el índice real que se emplea en la función de *hash* para acceder a la DMTE desde donde podrá hacerse la retransmisión en el caso que la PDU esté allí localizada. En el caso que la PDU no esté ya en la DMTE, se enviará la petición de retransmisión al conmutador previo, siguiendo un procedimiento idéntico al explicado en el párrafo anterior en que la pérdida era local.

11.8. RETRANSMISIONES DURANTE LOS TIEMPOS DE INACTIVIDAD

Uno de los aspectos básicos del rendimiento del protocolo TAP está en la idea intuitiva de aprovechar los estados de inactividad de los enlaces y de las fuentes con GoS para recuperar las pérdidas debidas a las congestiones. Ya adelantamos en el *Capítulo 8* los inconvenientes de las retransmisiones extremo-extremo en las que se basan protocolos tan exitosos como TCP que acaban pagando un precio demasiado elevado en cuanto a su rendimiento se refiere. Vimos entonces que TAP puede aportar importantes mejoras en goodput, porque las recuperaciones se realizan de forma local a los conmutadores que experimentan las pérdidas.

Además, otra ventaja del protocolo está en su posibilidad de no afectar a las transmisiones para atender las retransmisiones que, en muchas aplicaciones como sabemos, pueden ser perniciosas. Por tanto, en esta sección queremos demostrar que se satisface nuestra idea de partida, según la cual las retransmisiones pueden ser atendidas en los estadios en que las fuentes no generan tráfico o que haciéndolo, el enlace que emplean tiene disponibilidad de ancho de banda para soportarlas junto al tráfico normal. Vamos a comprobar, mediante fuentes ON/OFF, que las conexiones ATM pueden aprovechar muchos de esos estados de inactividad. Para ello debemos analizar y avaluar los tiempos de OFF agregado que nos quedan en la red después de multiplexar en un solo enlace *N* fuentes de tráfico. Realizaremos esa evaluación en primer lugar de forma conceptual para pasar luego a su estudio formulado, donde podremos comprobar la viabilidad para aplicar esta idea en TAP.

Por tanto, el protocolo analiza previamente la disponibilidad del tiempo de OFF agregado que queda en un enlace antes de atender las retransmisiones, ya que no tiene ningún sentido realizar retransmisiones cuando se sabe de antemano que éstas no tendrán éxito por el propio estado de sobrecarga de los enlaces. Cuando se detecta esta posibilidad TAP no pone en marcha el mecanismo de recuperación para no desaprovechar innecesariamente el rendimiento de la red.

11.8.1. MODELO DE UNA FUENTE ON/OFF Y POSIBILIDADES DE RECUPERACIÓN

Según la teoría de colas, el análisis del proceso de encolado es una parte fundamental para la evaluación del comportamiento de las redes. En el caso de ATM puede considerarse que una cola es una expresión matemática de la idea de contención de recursos. A las colas ATM llegan células, ráfagas o conexiones que necesitan un cierto servicio y, mientras son atendidos, esperan una determinada unidad de tiempo en una zona de almacenamiento (buffer, cola o línea de espera). El sistema de colas puede ser descrito según los siguientes aspectos: el patrón de llegadas de las conexiones, el patrón de servicio de las conexiones, el número de canales de servicio y la capacidad del sistema. Todos estos patrones han sido estudiados amplia y metódicamente en la literatura [1,2,3], aunque en el caso de ATM, suele descuidarse el parámetro de la sincronización en el análisis matemático de los buffers. Así, se asume que una célula es servida inmediatamente después de entrar en un buffer vacío, en lugar de esperar hasta el comienzo del siguiente slot libre. No obstante, nosotros tampoco deseamos entrar en estos niveles de detalle, ya que nuestro objetivo es mucho más generalista, en un intento por demostrar la viabilidad de TAP. No hemos de perder de vista que la mayor parte de propuestas de la arquitectura TAP están pensadas para optimizar el throughput de la red en general y, además, de forma particular, el mecanismo de retransmisiones mejora también el goodput, por lo que nos vamos a centrar en esta visión más amplia y para ello vamos a basarnos, como hemos dicho, en el modelo de fuentes ON/OFF.

Las fuentes ON/OFF pueden considerarse como un modelo basado en dos estados en los cuales la velocidad de llegada de células en cada estado es fija y el periodo de permanencia en cada estado puede ser exponencialmente, geométricamente o arbitrariamente distribuido. El modelo ON/OFF [1,2] lo usamos para caracterizar el tráfico ATM por conexiones unidireccionales y las fuentes ON/OFF a ráfagas nos van a permitir analizar el comportamiento de la pérdida de células. La *Figura 11.11* muestra este modelo como una fuente que: o bien envía datos (estado ON) durante un tiempo T_{on} a una velocidad CAR (*Cell Arrival Rate*); o bien permanece inactiva (estado OFF) sin producir células durante un tiempo T_{off}.

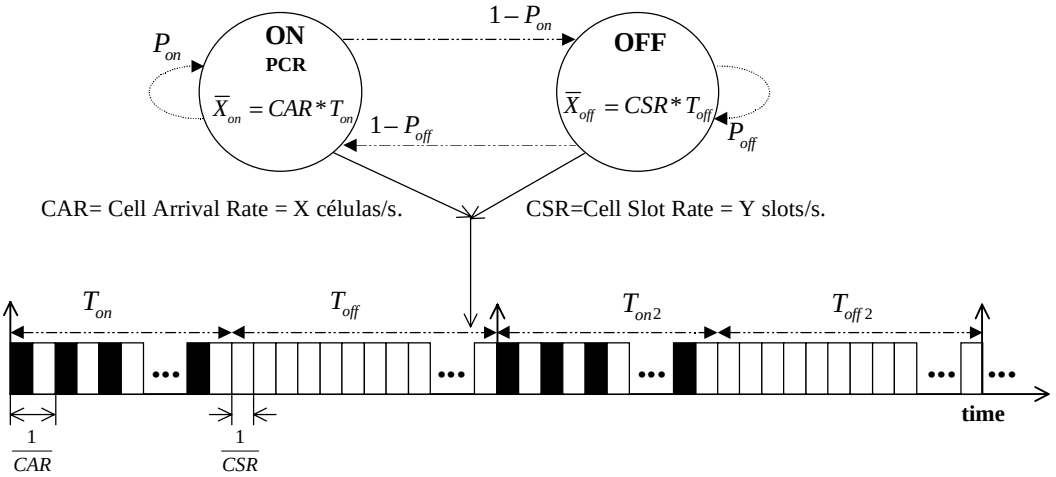

Figura 11.11. Patrón de generación de células para una fuente ON/OFF

Por tanto, en la fuente ON/OFF los procesos conmutan entre un estado de silencio (en que no se producen células) y un estado que produce una velocidad fija y determinada de células. Las fuentes ON/OFF con duraciones distribuidas como exponenciales negativas han sido frecuentemente estudiadas y aplicadas a tráfico de datos y como modelo general para tráfico a ráfagas en multiplexores ATM. Después de la llegada de cada célula se genera otra célula con probabilidad P_{on}, o bien la fuente cambia al estado de silencio OFF (*idle*) con una probabilidad $1\text{-}P_{on}$. De la misma forma, en el estado de silencio OFF la fuente genera otro slot de tiempo vacío con probabilidad P_{off}, o bien cambia al estado activo ON con una probabilidad $1\text{-}P_{off}$.

En lugar de ver el proceso de generación de células y el de slots de tiempo como procesos Bernoulli, simplemente los consideramos como procesos distribuidos geométricamente. Al final de un periodo de estancia en un estado concreto, los procesos conmutan al otro estado con probabilidad 1. Destacamos que las distribuciones geométricas de los estados de silencio y activos tienen diferentes bases de tiempos. Además, podemos comprobar en la *Figura 11.11* que en los estados de actividad pueden existir también slots vacíos, esto ocurre cuando el *CAR* es menor que el *CSR* (*Cell Slot Rate*) que es lo habitual.

Podemos calcular el valor de la unidad de tiempo de cada estado activo, o *CIT* (*Cell Inter-arrival Time*), como el inverso del *CAR*,

$$CIT = \frac{1}{CAR} \tag{1}$$

Así, la duración media de cada estado activo es,

$$T_{on} = \frac{1}{CAR}\overline{X}_{on} \tag{2}$$

donde $\overline{X}_{on}$ es el número medio de células que se produce en cada estado activo.

Por analogía con la expresión (2), en el estado de silencio, la unidad de tiempo es el inverso de *CSR*, por lo que la duración media del tiempo que la fuente permanece en el estado de silencio es,

$$T_{off} = \frac{1}{CSR}\overline{X}_{off} \tag{3}$$

donde en este caso $\overline{X}_{off}$ es el número medio de slots de células vacías que se producen en cada estado de silencio.

La expresión (2) nos permite calcular la media de células en un estado ON como el inverso de la probabilidad de salir del estado activo, aunque también lo podemos expresar como la velocidad de llegada de células *CAR*, multiplicada por el periodo de tiempo en que la fuente está activa. Así, podemos representar

este valor medio con la siguiente expresión, que calcula el número de células que se producen en cada estado activo de una fuente:

$$\overline{X}_{on} = \frac{1}{(1-P_{on})} = CAR * T_{on} \tag{4}$$

A través de la fórmula (3) podemos calcular también el número medio de slots de células que se producen en cada estado de silencio OFF como el inverso de la probabilidad de salir del estado *idle*. Como antes, podemos calcular este valor medio multiplicando el valor de *CSR* por el tiempo en que la fuente permanece en inactividad. De este modo, podemos expresar este valor medio con la siguiente expresión que indica el número medio de slots que se producen en cada estado de silencio de una fuente:

$$\overline{X}_{off} = \frac{1}{(1-P_{off})} = CSR * T_{off} \tag{5}$$

De esta forma, podemos generalizar el modelo de las fuentes ON/OFF para distribuciones arbitrarias, tanto del número de células generadas en el estado activo, como del número de slots vacíos generados en los estados de silencio.

Podemos aplicar todo esto a un ejemplo concreto como es la supresión de los silencios telefónicos (en las cuales no se transmitirían células durante los periodos en los que los interlocutores están en silencio). Estudios empíricos [1] demuestran que se pueden considerar los siguientes tiempos medios en los estados de ON y OFF

$$\begin{aligned} T_{on} &= 0{,}96s. \\ T_{off} &= 1{,}69s. \end{aligned} \tag{6}$$

Seguidamente vamos a aplicar toda la formulación anterior a diversos casos que nos permitirán argumentar nuestros objetivos. Supongamos el caso de una fuente con un flujo de llegadas periódicas a una velocidad de 64 Kbps sobre un enlace de 155,52 Mbps. Si expresamos este flujo en forma de células por segundo tendremos el valor de *CAR* según la siguiente expresión,

$$CAR = 64Kbps = \frac{\left(\frac{64.000}{8}\right)}{48} = 167 \tag{7}$$

Tenemos, por tanto, que la velocidad de llegada de células para esta fuente es de 167 células por segundo. De forma parecida podemos calcular el valor del *CSR* si suponemos usar un enlace de 155,52 Mbps, con la siguiente fórmula que expresa el número de slots por segundo que se producen en cada estado de OFF,

$$CSR = \frac{\left(\frac{155.520.000}{8}\right)}{53} = 366.792 \tag{8}$$

Partiendo del valor obtenido en (8) podemos calcular el *Tiempo de Servicio por Célula* (*TSC*) lo que arroja un resultado de 2,726 μseg. como tiempo de proceso de cada célula.

Podemos afinar el resultado de (8) si tenemos en cuenta que en ATM se genera una células *OAM* (*Operation And Management*) cada 27 células, y obtendremos el número total de slots de células válidos que se generan en cada estado OFF,

$$\frac{26}{27} 366.792 = 353.208 \tag{9}$$

En el caso de (9) podemos calcular el nuevo *TSC* que es de 2,831 μseg para cada uno de los slots de células generadas.

Con los valores de los tiempos de ON y OFF fijados en (6) podemos calcular el número medio de células producidas en un estado activo:

$$\overline{X}_{on} = CAR * T_{on} = 167 * 0{,}96 = 160 \tag{10}$$

De forma similar, el número medio de slots vacíos generados en un estado de silencio es,

$$\overline{X}_{off} = CSR * T_{off} = 353.208 * 1{,}69 = 596.921 \tag{11}$$

Podemos calcular también las probabilidades de paso P_{on} y P_{off} entre estados ON y OFF respectivamente. Sabemos que,

$$\overline{X}_{on} = \frac{1}{(1 - P_{on})} = 160 \tag{12}$$

por tanto,

$$P_{on} = 1 - \left(\frac{1}{160}\right) = 0{,}99375 \tag{13}$$

Del mismo modo,

$$\overline{X}_{off} = \frac{1}{(1 - P_{off})} = 596.921 \tag{14}$$

por tanto,

$$P_{off} = 1 - \left(\frac{1}{596.921}\right) = 0{,}99999832 \tag{15}$$

Según todo esto, llegamos a las siguientes conclusiones:

- En el estado de silencio se generan 353.208 slots por segundo, aunque la media de slots generados es de 596.921. Además, se permanece en el estado de silencio durante 1,69 seg., y la probabilidad de continuar en él es de 0,99999832, por lo que se cambia del estado de silencio al de actividad con una probabilidad de 0,00000168.
- En el estado de actividad se generan células a una velocidad de 167 por segundo. Como se permanece en el estado de actividad durante 0,96 seg., el número medio de células en cada estado activo es de 160 células y la probabilidad de permanecer en este estado es de 0,99375, mientras la probabilidad de cambiar al estado de silencio es de 0,00625.
- Sin considerar que en los estados activos puede haber slots vacíos, las 160 células de cada estado ON representan el 0,000268 % de los slots que se producen en el T_{off}. Para realizar los cálculos se han usado valores fijos y concretos de T_{on} y T_{off} que pueden ser diferentes en otras clases de servicio, pero se puede comprobar que, con otros valores de T_{on} y T_{off} más próximos, los resultados finales son muy parecidos. En realidad, lo que nos sirve para apoyar nuestra idea es la gran diferencia entre el valor de *CSR* del enlace con respecto al *CAR* de la fuente ON. Cada vez que se entra en un estado OFF se generan muchos más slots de células por segundo que las células realmente generadas en cada estado ON. Esto es lo habitual en las redes ATM donde los conmutadores se caracterizan por su gran potencia de conmutación para poder soportar las necesidades de cada enlace que, generalmente, no son completamente aprovechadas por las fuentes.

Podemos realizar planteamientos similares para otro ejemplo de fuente cuatro veces más rápida que la anterior (512 Kbps) sobre un enlace también de 155,52 Mbps . En este escenario los resultados obtenidos son los siguientes:

$$CAR = ((512.000 / 8) / 48) = 1.334 cps$$
$$CSR = 353.208 sps$$
$$\overline{X}_{on} = CAR * T_{on} = 1.334 * 0,96 = 1.280 c.$$
$$\overline{X}_{off} = CSR * T_{off} = 353.208 * 1,69 = 596.921 slots$$

Supongamos ahora un tercer escenario, donde la fuente es de 1,5 Mbps (que podría ser adecuado para tráfico multimedia) sobre el mismo enlace de 155,52 Mbps. Este escenario se ha representado en la *Figura 11.12* donde podemos observar la confluencia de la fuente ON/OFF en un conmutador AcTMs con puertos de ancho de banda de 155,52 Mbps. La *Figura 11.12* representa también los resultados que demuestran que el *CSR*, para una sola fuente, se mantiene en un valor muy superior al de *CAR* lo que permite disponer de tiempos de inactividad suficientes como para atender las solicitudes de retransmisión.

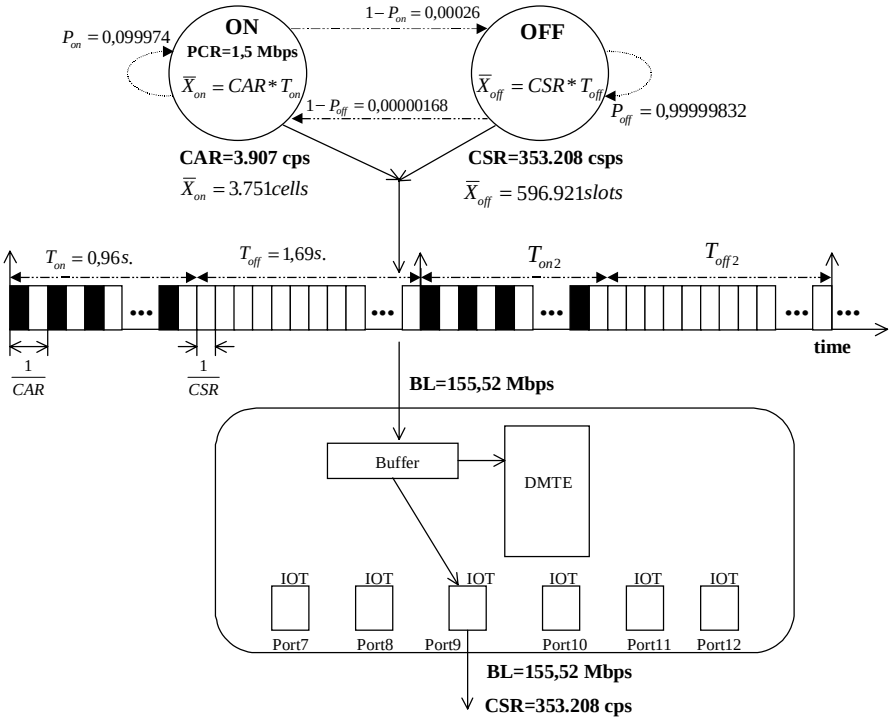

Figura 11.12. Patrón de llegada de células para una fuente ON/OFF concreta en un AcTMs

En el siguiente punto estudiamos el efecto del T_{off} agregado con *n* fuentes; no obstante, supongamos ahora el caso de un conmutador ATM real tipo LE-155 con 16 ports de 155,52 Mbps cada uno y una potencia de conmutación de 2,488 Gbps a la salida del multiplexor interno del conmutador:

$$CAR = ((155.520.000 / 8) / 48) = 405.000 cps$$
$$CSR = ((2.488.320.000 / 8) / 53) * (26 / 27)) = 5.651.322 sps$$
$$\overline{X}_{on} = CAR * T_{on} = 405.000 * 0,96 = 388.800 c.$$
$$\overline{X}_{off} = CSR * T_{off} = 5.651.322 * 1,69 = 9.550.735 slots$$

En este caso, si suponemos tener las 16 fuentes simultáneamente a 155,52 Mbps se generarían 16*405.000 = 6.480.000 células que son también menores que los slots que pueden producirse en cada estado de OFF, lo que justifica la posibilidad de realizar retransmisiones durante las diferencias entre los tiempos de actividad y de silencio como veremos en la siguiente sección.

Los ejemplos anteriores demuestran conceptualmente la idea intuitiva de que es factible la realización de retransmisiones en los estados de silencio en el caso de emplear una sola fuente. Hemos partido de valores empíricos y concretos para T_{on} y T_{off}, pero podemos ver cómo realizando leves cambios en estos valores sigue existiendo un amplio margen para poder realizar las retransmisiones en los estados de inactividad.

11.8.2. DISPONIBILIDAD DE T_{OFF} AGREGADO CON N FUENTES ON/OFF

En los escenarios anteriores hemos podido comprobar las posibilidades de retransmisión aprovechando los estados de inactividad de una única fuente. Parece evidente que las posibilidades de retransmisión empiezan a disminuir a medida que introducimos más fuentes en los escenarios, porque a un conmutador llegan más conexiones que van a terminar siendo multiplexadas sobre un mismo puerto de salida. En este caso parece que el tiempo T_{off} total de un enlace será compartido por varias fuentes, por lo que es importante analizar qué ocurrirá en esta situación. Pues bien, podemos generalizar el modelo ON/OFF de dos estados a N estados con velocidades fijas en cada uno de los estados. Estos escenarios multi-estado (llamados procesos determinísticos modulados) pueden usarse para modelar un número concreto de N fuentes ON/OFF multiplexadas sobre un mismo enlace:

- Si los tiempos de permanencia siguen una distribución arbitraria, el proceso resultante es llamado *GMDP* (*Generally Modulated Deterministic Process*).
- Si las duraciones de los estados son exponencialmente distribuidas, el proceso es llamado *MMDP* (*Markov Modulated Deterministic Process*). En este caso, cada estado produce un número de células distribuidas geométricamente durante cada periodo de permanencia. Esto es así porque, habiendo generado la llegada *i*, se genera la llegada *i+1* con una probabilidad dada por la posibilidad de que el tiempo de permanencia no acabe antes del tiempo de la siguiente llegada. Esta probabilidad es una constante si los periodos de permanencia son distribuidos exponencialmente debido a la propiedad de "no-memorización" de las distribuciones exponenciales negativas.
- Puede evitarse el restringir el modelo para tener una velocidad constante de llegadas en cada estado. Si el proceso de llegada por estado sigue una Poisson, tendremos una *MMPP* (*Markov Modulated Poisson Process*) que puede ser muy útil para representar un proceso de fuentes agregadas.

Pero en nuestro caso, puede tener mayor interés otro punto de vista más práctico para el estudio de múltiples fuentes ON/OFF que confluyen en un solo buffer o puerto de salida. Supongamos tener N fuentes ON/OFF idénticas que envían células a un buffer de capacidad de servicio de C células por segundo y de un tamaño fijo y finito de T células. T_{on} y T_{off} son las duraciones medias en los estados ON y OFF respectivamente como ya vimos antes. *CAR* es de nuevo la velocidad de llegada de células por segundo en el estado activo pero, en este caso, introducimos el parámetro *MCR* (*Mean Cell Rate*) que expresa la velocidad media de llegadas de células en cada fuente:

$$MCR = CAR\left(\frac{T_{on}}{T_{on} + T_{off}}\right) \tag{16}$$

En este caso, calculamos la probabilidad de que la fuente esté en el estado activo, o lo que podemos denominar como el factor de actividad (*FA*) de cada fuente como,

$$FA = \frac{MCR}{CAR} = \frac{T_{on}}{T_{on} + T_{off}} \tag{17}$$

Podemos ver aquí un aspecto interesante para demostrar nuestra propuesta en el caso de N fuentes ON/OFF, que es el estudio de un parámetro que nos permita saber cuántas veces cabe (o encaja) la velocidad *CAR* de las N fuentes de tráfico en la capacidad de servicio C del buffer y que podemos denominar como número máximo de fuentes privilegiadas (*FP*), a las que podemos garantizar que dispondrán de suficiente tiempo de OFF agregado en el enlace para poder atender las retransmisiones en el caso de congestiones. Denotamos este parámetro con la expresión,

$$FP = \frac{C}{CAR} \tag{18}$$

La *Figura 11.13*, con *N* fuentes ON/OFF idénticas operando independientemente, representa los conceptos que acabamos de comentar y aplicados al tercer escenario que hemos representado en la *Figura 11.12*. Podemos comprobar cómo en este caso *FP*=90,4 fuentes, cada una de ellas de 1,5 Mbps sobre un enlace de 155,52 Mbps y con un factor de actividad *FA* del 36%.

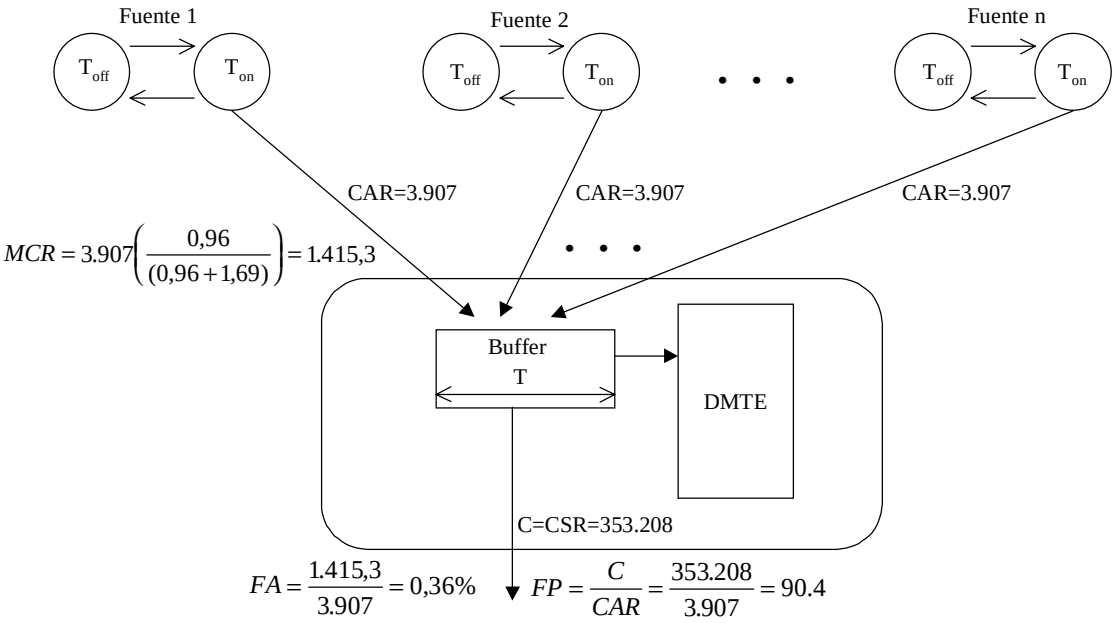

Figura 11.13. *Modelo de múltiples fuentes ON/OFF*

El valor de *FP* puede ser un valor no entero. Si lo redondeamos por encima tendremos el número mínimo de fuentes a partir del cual se puede producir la congestión del buffer. Si redondeamos el valor de *FP* por debajo tendremos el número máximo de fuentes del sistema para no producir congestiones del buffer, o también aquel número máximo de fuentes a las que podremos garantizar las retransmisiones por disponer de suficiente tiempo de inactividad agregado.

En la arquitectura que proponemos tenemos un solo buffer de entrada para las fuentes de tráfico y podemos considerar que la salida del buffer es el punto de multiplexación de todas las fuentes. La multiplexación de las fuentes de entrada debe ser, por tanto, menor o igual que la capacidad de servicio del buffer para que no se produzca la congestión a la salida de éste que, en nuestro caso, consideramos como el puerto de salida del conmutador en el que confluyen las fuentes de entrada para ser multiplexadas.

Según los apartados anteriores, para nosotros es importante aprovechar los tiempos de inactividad de los enlaces de salida en los conmutadores sobre los que se produce la multiplexación de diversas fuentes de entrada. La *Figura 11.14* aporta esta visión intuitiva aprovechando las fuentes ON/OFF que venimos usando. En esta figura podemos observar los modelos de llegada de tres fuentes con sus correspondientes tiempos de actividad y de silencio. Se presenta también el modelo que obtendríamos a la salida de un enlace en el que se multiplexan las tres fuentes, de forma que podemos observar cómo los tiempos T_{on} de cada una de las fuentes se reflejan a la salida. En la salida el tiempo de inactividad es menor, ya que la multiplexación de las tres fuentes se hace a costa de reducir el tiempo de inactividad del enlace. Precisamente, nuestro objetivo es el de atender las retransmisiones en el caso que el T_{offAg} agregado que queda en los enlaces así lo permita. La *Figura 11.14* nos muestra, por tanto, que el T_{on} a la salida es la suma de los T_{on} de las entradas, por lo que el T_{offAg} agregado que queda disponible a la salida para las retransmisiones es el T_{off} original al que hay que restar la suma de los T_{on} de las entradas. De este modo, antes de realizar la solicitud de una retransmisión es necesario evaluar el tiempo de inactividad que tenemos disponible en un enlace, ya que puede ocurrir que el T_{offAg} agregado sea menor que el T_{on} de la fuente cuya retransmisión desea realizarse. En esta situación la retransmisión no será solicitada, ya que no haríamos más que sobrecargar la red con una solicitud que no tendrá cabida en el enlace y que, con seguridad, acabará provocando lo que en las simulaciones hemos denominado como fallos de memoria DMTE.

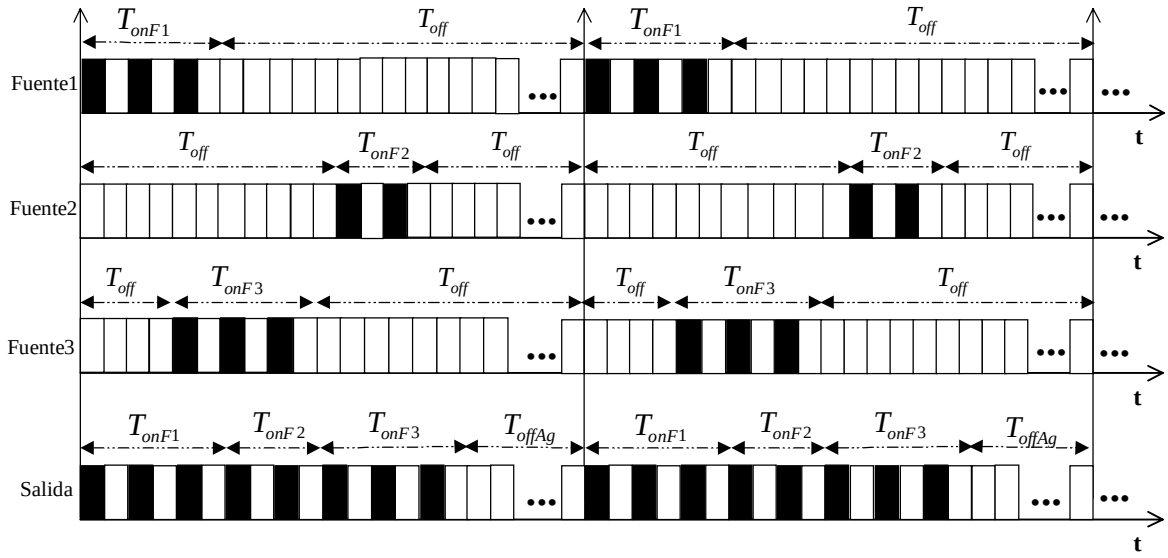

Figura 11.14. *Multiplexación de tres fuentes de entrada en un puerto de salida con T_{offAg} agregado*

A continuación vamos a formalizar estas reflexiones para justificar nuestros planteamientos. Para ello fijamos la siguiente notación:

T_{onE} ; Es el tiempo de actividad a la entrada

$\overline{T}_{onE}$; Es el tiempo de actividad medio a la entrada

T_{offS} ; Es el tiempo de silencio a la salida

$\overline{T}_{offS}$; Tiempo de silencio medio a la salida.

PCR_E ; PCR a la entrada

PCR_S ; PCR a la salida

N ; Representa el número de fuentes

$\overline{R}_E$; Velocidad media de llegada de células a la entrada (CAR medio a la entrada).

$\overline{R}_S$; Velocidad media de llegada de células a la salida (CAR medio a la salida).

En el caso de una sola fuente, el tiempo de actividad medio se representa en general según la siguiente expresión,

$$\overline{T}_{on} = \frac{\overline{R}}{PCR}\overline{T}_{off} \tag{19}$$

Si suponemos disponer de *N* fuentes de entrada, podemos calcular el tiempo de actividad a la salida de un conmutador sobre el que se multiplexan las *N* fuentes,

$$\overline{T}_{onS} = \frac{N\overline{R}_S}{PCR_S}\overline{T}_{offS} \tag{20}$$

De (20) podemos obtener entonces el tiempo de inactividad agregado disponible según,

$$\overline{T}_{offS} = \frac{PCR_S}{N\overline{R}_S}\overline{T}_{onS} \tag{21}$$

Considerando la expresión como valor a la entrada de un conmutador, y despejando la velocidad media R_E de llegada de células a la entrada (variante de entrada en la ecuación (20)), y sustituyéndola en (21) obtendremos,

$$\overline{T}_{offS} = \frac{1}{N}\frac{PCR_S}{PCR_E\frac{\overline{T}_{onE}}{\overline{T}_{offE}}}\overline{T}_{onS} = \frac{1}{N}\frac{PCR_S}{PCR_E}\frac{\overline{T}_{onS}}{\overline{T}_{onE}\Big/\overline{T}_{offE}} \tag{22}$$

Si ajustamos (22) podemos llegar a la siguiente expresión

$$\frac{\overline{T}_{offS}}{\overline{T}_{onS}} = \frac{1}{N}\frac{PCR_S}{PCR_E}\frac{\overline{T}_{offE}}{\overline{T}_{onE}} \tag{23}$$

Agrupando los tiempos en (23) obtenemos,

$$\frac{\overline{T}_{onS} + \overline{T}_{offS}}{\overline{T}_{onE} + \overline{T}_{offE}} < \frac{1}{N}\frac{PCR_S}{PCR_E} \tag{24}$$

Si consideramos que $PCR_S = N * PCR_E$, podemos comprobar cómo las medias de tiempos a la entrada coinciden con la suma de los tiempos medios de entrada,

$$\overline{T}_{onS} + \overline{T}_{offS} = \overline{T}_{onE} + \overline{T}_{offE} \tag{25}$$

No obstante, nuestro objetivo principal es el de poder expresar el valor del tiempo de OFF a la salida y en función de las entradas, lo que nos servirá para estimar el valor agregado de que disponemos para la retransmisión. Así, podemos expresar el valor medio de llegada de células de una nueva forma,

$$\overline{R} = \frac{PCR_E}{T_{onE} + T_{offE}} \Rightarrow T_{offE} = \frac{PCR_E}{\overline{R}} - T_{onE} \tag{26}$$

Usando (26) podemos expresar (24) despejando el valor que nos interesa, y sustituyendo $T_{onS} = NT_{onE}$,

$$\overline{T}_{offS} = \frac{1}{N}\frac{PCR_S}{PCR_E}(\overline{T}_{onE} + T_{offE}) - T_{onS} = \frac{1}{N}\frac{PCR_S}{PCR_E}(\overline{T}_{onE} + \frac{PCR_E}{\overline{R}} - T_{onE}) - NT_{onE} \tag{27}$$

El reajuste de (27) nos lleva a

$$\overline{T}_{offS} = \frac{PCR_S}{N\overline{R}} - NT_{onE} \tag{28}$$

Para poder atender las retransmisiones, el tiempo de OFF debe ser superior a 0, y para que esto sea así, de (28) deberemos tener,

$$\frac{PCR_S}{N\overline{R}} > NT_{onE} \tag{29}$$

De este modo podemos llegar a la expresión,

$$PCR_S > N^2 T_{onE} \overline{R} \tag{30}$$

Que, debidamente ajustada, acaba dando lugar a,

$$PCR_S > N^2 PCR_E \frac{T_{onE}}{T_{onE} + T_{offE}} \tag{31}$$

La fórmula (31) expresa, por tanto, una referencia válida del límite que debe tener el PCR en la salida, expresado en los valores conocidos de la entrada, para poder disponer de un T_{offAg} agregado suficiente para garantizar que la retransmisión es factible en esa situación.

11.9. IMPLICACIONES SOBRE LA SEÑALIZACIÓN

Como es sabido, ATM es una tecnología orientada a la conexión, por lo que es necesario disponer de un protocolo de señalización encargado de la configuración, modificación, control y realización de cada una de las conexiones. Los parámetros que identifican cada conexión (VPI, VCI, ports, PCR, GoS, QoS, etc.) son acordados entre los dos extremos de la comunicación y con la red que debe comprobar si dispone de suficientes recursos y si no existe incompatibilidad entre los extremos que impida la comunicación. El protocolo de señalización es el encargado de realizar estas funciones [6] y, para ello, debe contar con un proceso en el que estén implicados los tres elementos que deben acordar los citados parámetros. Cuando la red está compuesta de múltiples nodos, éstos deberán acordar también los parámetros entre ellos [7].

Aunque los protocolos UNI y NNI son muy parecidos, existen diferencias apreciables como, por ejemplo, que en el caso de NNI debe controlarse la multiplexación en la situación habitual en que se soporten múltiples conexiones en un enlace. Así, el protocolo de señalización ha sido estandarizado por ITU-T en su recomendación Q.2931[8], y por ATM Forum en UNI 4.x [6] que no especifica ningún estándar para NNI. Las normas ITU-T y ATM Forum son muy parecidas y, aunque tienen ligeras diferencias, ambas requieren de un canal virtual de señalización propio y separado de los canales de datos (*out band*) para la negociación de los parámetros entre los usuarios y la red. Las dos recomendaciones establecen los mensajes definidos para realizar todas las negociaciones iniciales y el control durante todo el tiempo que dure la comunicación.

Es interesante destacar también las propuestas de señalización pensadas para servicios específicos, como SSCOP (Service-Specific Connection-Oriented Protocol) [9], que se ha propuesto situado en la pila de protocolos ATM sobre la capa AAL-5 y se encarga de aspectos como el control de flujo, la corrección de errores basándose en un mecanismo de retransmisiones, etc..

No es nuestro objetivo desarrollar los aspectos relativos a la señalización en la red; sin embargo, debemos destacar que la aplicación del protocolo TAP implicaría emplear la señalización estándar por un lado, y por otro habría que adaptar ligeramente algunos de los aspectos de la misma para disponer de información relativa a la red que es controlada por estos protocolos específicos. Así, y aunque no se ha especificado expresamente en los algoritmos estudiados en este capítulo, en las funciones y procedimientos donde se realizan las labores de negociación y liberación de la conexión, en las transferencias entre capas, en la actualización de las Tablas de E/S, etc., intervienen también los protocolos de señalización.

Algunos de los aspectos de la señalización de mayor interés para alcanzar nuestros objetivos están relacionados con la identificación de los canales de comunicación, tanto en UNI, como en NNI, ya que de esta identificación (VPI, VCI, ports) depende el mecanismo de recuperación propuesto en nuestras investigaciones. Según esto, las funciones de negociación de la conexión, de establecimiento y liberación de la misma, los procedimientos de creación de índices -tanto en las Tablas de E/S como en la memoria DMTE- y la propia transferencia de células BRM como resultado de una solicitud de retransmisión nos exigen acceder al canal de señalización para identificar cada una de las conexiones implicadas en las transferencias que han negociado la GoS con la red. Estos accesos nos permitirán conocer los VPI, los VCI y los puertos de entrada y de salida de cada una de las conexiones que deseamos controlar para poder garantizarles un servicio privilegiado con respecto al resto de conexiones que no lo demanden.

11.10. CONCLUSIONES

En este capítulo hemos detallado los aspectos de mayor interés del conjunto de algoritmos que constituyen el protocolo TAP. Hemos centrado nuestra atención en destacar la visión que tienen de TAP, tanto los nodos terminales de la comunicación, como los conmutadores activos que soportan el SMA-TAP. Por esto hemos presentado también los algoritmos que implementan cada uno de los agentes software del SMA, describiendo sus puntos de comunicación, así como la relación con los elementos hardware de la arquitectura que fueron comentados en el capítulo anterior.

Hemos modelado mediante fuentes ON/OFF los patrones de llegada de células a los conmutadores para formalizar la idea intuitiva de aprovechar los estados de inactividad de las fuentes y de los enlaces para atender las solicitudes de retransmisión cuando uno de los conmutadores esté experimentando congestiones. Esta formalización nos permite comprobar cómo es factible aprovechar los periodos de silencio para recuperar las pérdidas, sin afectar por ello ni al goodput de la red, ni tampoco al resto de las fuentes que puedan estar compartiendo un mismo conmutador o enlace.

Por último, hemos comentado brevemente algunos de los aspectos relativos a la señalización y control que están relacionados con TAP, de forma que vamos a depender de los protocolos estándares de la propia tecnología ATM para alcanzar nuestros objetivos.

REFERENCIAS

[1] J. M. Pitss, "Introduction to ATM. Design and Performance", *Ed. Wiley*, (1997).

[2] A.L. Roginsky, L.A. Tomek and K.J. Christensen, " Analysis of ATM Cell Loss for Systems with on/off Traffic Sources," *IEE Proc. Commun.* Vol. 144. No. 3, pp. 129-134, (1997).

[3] Sebastián Galmés Obrador, "Algoritmos de captura a gran escala de fuentes de tráfico ATM," *Memoria de tesis doctoral*, (1998).

[4] K. Kvols and S. Blaabjerg, "Bounds and approximations for the periodic on/off queue with applications to ATM traffic control," *INFOCOM'92*, pp. 487-494, (1992).

[5] K. Sohraby, "Onthe theory of general on-off sources with applications in high-speed networks," *INFOCOM'93*, pp. 401-410, (1993).

[6] ATM Forum, "Traffic Management Specification Version 4.0," *ATM Forum Technical Committee, ATM Forum Document af-tm-0056.000*, (April 1996).

[7] M. de Prycker, "Asynchronous Transfer Mode. Solution for Broadband ISDN (3ª Ed.)," *Ed. Prentice Hall*, (1995).

[8] ITU-T, Q.2931, "BISDN User Network Interface Layer 3 Protocol," (1994).

[9] ITU-T, Q.2110, "BISDN Signalling ATM Adaptation Layer (SAAL). Service Specific Connection Oriented Protocol SSCOP," (1994).

CAPÍTULO 12

SIMULADOR DE TAP Y ANÁLISIS DE RESULTADOS

12.1. INTRODUCCIÓN

Podemos considerar que un protocolo es un módulo software que implementa una serie de especificaciones de comunicación para ofrecer un conjunto de servicios a las aplicaciones que los requieran. Desde el punto de vista más tradicional los podemos ver como un conjunto de normas o reglas que rigen la comunicación dentro de una tecnología de comunicaciones, generalmente estratificados en un conjunto de capas, que acaban dando lugar a una pila de protocolos que es atravesada por la secuencia de mensajes (datos y control) que genera una sesión de comunicación en la red.

En nuestro caso, TAP aporta una serie de servicios de comunicación que se han implementado usando múltiples algoritmos distribuidos que interactúan entre sí, como ya hemos tenido la ocasión de comprobar en capítulos previos. El diseño de protocolos de comunicación, como sabemos, no es una tarea trivial, pero la implementación robusta con buen rendimiento de la red y con algoritmos interrelacionados en un SMA se convierte en una tarea realmente compleja. Como hemos podido comprobar en el *Capítulo 11*, la implementación del protocolo TAP, como otros tantos protocolos, divide su actividad en dos importantes aspectos, por un lado el flujo de información a través de la pila de protocolos y a lo largo de toda la red, y por otro, todo lo relacionado con el mantenimiento de la información de estado y control en el mantenimiento del propio protocolo.

Muchas son las ventajas que podemos encontrar en los SMA, pero en general podemos destacar las aportadas en [1] para justificar nuestra decisión de proponer un SMA como forma de solventar los problemas identificados en la tecnología ATM: "*La razón por la que los agentes ofrecen un excitante paradigma para la resolución de problemas es debida a las muchas ventajas aportadas por los sistemas basados en agentes. Los agentes ofrecen un paradigma para la programación remota inteligente. Los SMA ofrecen una vía por la que relajar los inconvenientes de la centralización, planificación y control secuencial para ofrecer sistemas que son descentralizados, emergentes y concurrentes. Además, ayudan a reducir tanto los costes software como hardwar,e y ofrecen soluciones rápidas a problemas aprovechando las ventajas del paralelismo.*"

Aunque en los dos o tres últimos años el campo de los SMA ha experimentado bastante actividad, aun hoy puede decirse que los diseñadores e implementadores de sistemas multiagente y, sobre todo en el ámbito de las telecomunicaciones, son bastante escasos. Disponemos por tanto de una interesante excusa para investigar en materia de ingeniería de protocolos basada en agentes software. Existen en la actualidad múltiples SMA ya implementados y diversas herramientas para la construcción de agentes y de SMA, unas basadas en Java, otras en otros lenguajes, pero las propuestas en el ámbito de los protocolos de comunicación son escasas y poco desarrolladas. En el caso de la tecnología ATM la escasez es aún mayor por lo que nos hemos propuesto buscar una solución a los problemas identificados mediante la especificación de un SMA. No obstante, destacamos que el SMA-TAP es la excusa para investigar en este atractivo campo, pero no hemos de perder de vista que nuestro principal objetivo es buscar una solución válida al problema planteado por las congestiones en los conmutadores ATM, sea solventado por los métodos más tradicionales de la

ingeniería de protocolos, o a través de técnicas inspiradas en SMA. Desde este punto de vista llamamos la atención sobre el hecho que el SMA-TAP se propone como un primer prototipo que ha de ser refinado en investigaciones futuras y, como tal, lo identificaremos en este capítulo y en el siguiente donde comentaremos las líneas de continuidad de nuestras investigaciones en esta materia.

En lo que respecta al flujo de la información, los protocolos gestionan el tráfico que fluye en dos sentidos: desde los usuarios hacia la red, y desde ésta hasta los propios usuarios. Esto lo hemos podido comprobar en el *Capítulo 11*, donde el protocolo TAP en los extremos de la comunicación hace de interfaz entre las aplicaciones y las placas de red de los usuarios. Entre TAP y la red se dispone de la pila de protocolos estándar ATM donde también hemos ampliado las características de la capa AAL-5. No obstante, estas capas estándares del modelo arquitectónico ATM realizan todas las labores del estándar, ya sea en cuanto al flujo de información, o en cuanto al mantenimiento de la información de estado, tal como hemos asumido en los algoritmos ya presentados. En cuanto al flujo de información en los AcTMs también hemos podido ver cómo los agentes intercambian información entre ellos y controlan a la vez el flujo de los datos de usuario. Como cada uno de los agentes está jerárquicamente uno sobre otro, esto permite que sean fácilmente configurables. El conjunto de relaciones entre los algoritmos que intervienen en el flujo de información nos permite establecer un mapa de dependencias y relaciones entre los algoritmos que da lugar a lo que llamamos la tabla del protocolo. En esta tabla las filas pueden ser los protocolos o algoritmos y las columnas representan las relaciones de cada algoritmo con sus vecinos. Como cada algoritmo (o agente) puede ser más o menos dependiente de otros algoritmos (o agentes) la tabla representa las conexiones de unos con otros y, por tanto, sus servicios y relaciones para lo que será necesario disponer de un interfaz genérico inter-protocolo, inter-algoritmos o inter-agentes. En nuestro prototipo con Java, es realizado a través de threads de comunicación inter-agentes que se encargan de mantener la comunicación y las relaciones entre cada uno de los algoritmos. Para cada flujo de datos entrante o saliente en un algoritmo, éste debe identificar el siguiente algoritmo y reenviarle al mensaje para que éste lo procese.

En lo que respecta a la información de estado hay que destacar que la implementación de protocolos requiere mantenerla para cada una de las conexiones establecidas. Esta información de estado suele ser de dos tipos: por un lado información global de toda la implementación del protocolo, y por otro lado, información específica que corresponde a cada una de las conexiones particulares establecidas por los usuarios. En el caso de TAP puede considerarse como información global a todo el protocolo, la identificación de los nodos activos que intervienen, los circuitos virtuales establecidos, o los tamaños de DMTE que intervienen en la red. Podemos considerar como información específica de cada comunicación, todas aquellas que se emplean para caracterizar cada fuente en el proceso de establecimiento de la conexión (grado de GoS, ToS, Ton, Toff, etc.) o, por ejemplo, el control de los PDUid que son propios de cada conexión extremo-extremo.

Debemos señalar también por otro lado las informaciones de control propias de la red y que están también relacionadas directamente con TAP, ya que esta información de control es la que, a través de la información de estado (tanto general como específica), permitirá controlar el flujo de la información de datos. A parte de los aspectos propios de la señalización que intervienen en la comunicación, en nuestro caso podemos también considerar como tal el propio mecanismo de recuperación de pérdidas basado en la generación de BRM como ya sabemos.

La *Figura 12.1* muestra un esquema del agente RCA que es uno de los agentes más complejos del SMA-TAP por las interrelaciones que tiene con otros agentes y con las estructuras de datos propias de los nodos AcTMs. Podemos observar cómo el agente dispone de un control de métodos remotos por los cuáles se pueden invocar desde otros agentes determinadas acciones. Por ejemplo, RCA dispone de un método remoto pensado para atender las peticiones de retransmisión que le llegan desde otros agentes RCA situados en otros nodos activos. Cuando este método es invocado el agente encola la petición en la cola de peticiones que, mediante un thread, es atendida siguiendo el ciclo de vida de éste y según las consideraciones que son explicadas en los *Capítulos 10* y *11*. Del mismo modo se incorporan otros métodos remotos para acciones concretas como la atención de retransmisión proveniente desde el algoritmo EPDR que forma parte del agente CCA y que se desencadena cuando la congestión es local al nodo que implementa el propio agente RCA. Una vez que el control de métodos remotos pasa la notificación a la cola de peticiones, se realiza una notificación al agente que solicita la retransmisión como un mecanismo de confirmación de que el método remoto invocado ha sido atendido.

En este capítulo justificaremos la elección del lenguaje Java para la implementación de protocolos de comunicación y de SMA. Las ventajas destacadas son las que nos han animado a elegir este lenguaje como herramienta para el desarrollo de nuestro prototipo de simulador de TAP (TAPS). En el siguiente punto comentamos la metodología y decisiones de diseño tomadas para el desarrollo de TAPS, poniendo especial atención en el SMA. A continuación se realiza la especificación de las clases, métodos y atributos del SMA-TAP,

sin ánimo de ser extensivos para no distraer la atención sobre el verdadero objetivo de esta tesis. Los dos últimos apartados del capítulo tienen por objetivo analizar algunos de los resultados más significativos obtenidos en las simulaciones realizadas. En primer lugar se presentan varios escenarios de simulación donde podemos comprobar las posibilidades de recuperación de PDU congestionadas, y también demostramos que se cumple la idea intuitiva de aprovechas los estados de inactividad de los enlaces para abordar las retransmisiones. Después se presta atención a la gestión de las colas de entrada, por ser uno de los aspectos de mayor complejidad. El capítulo termina con un apartado de conclusiones.

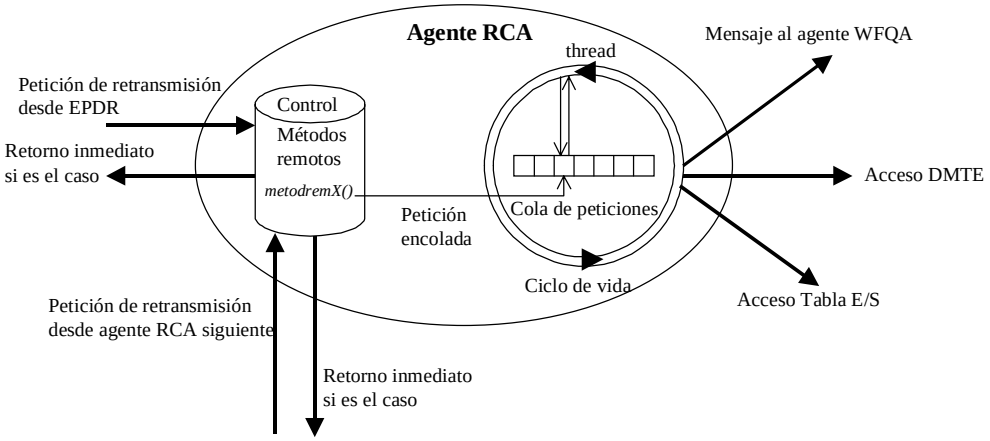

Figura 12.1. *Arquitectura del agente RCA y método remoto para la gestión de mensajes del agente*

12.2. IMPLEMENTACIÓN DE PROTOCOLOS Y SMA CON JAVA

En los últimos años se han propuesto varios lenguajes de programación para la implementación de agentes móviles. Los lenguajes más extendidos realizan diferentes decisiones de diseño como son la determinación de qué componentes de estado de un programa pueden migrar de un nodo a otro. En Java [2], sólo puede migrar el código del programa y el estado no puede ser transportado con el programa. Por otro lado, *Obliq* [3], los valores de función de *first-class* (*closures*) pueden migrar; los *closures* consisten en el código de programa, junto con el entorno que liga variables a valores o posiciones de memoria. Por último, en *Telescript* [4] las funciones no son valores *first-class*, y este lenguaje ofrece operaciones especiales que permiten migrar a los procesos autónomamente. Estos lenguajes también se diferencian en la forma en que transportan objetos entre agentes. Cuando un proceso o un *closure* migra, puede transportar todos los objetos (datos mutables), o dejar los objetos y transportar por la red las referencias a los objetos. Java no permite hacer esto ya que sólo permite la migración del código de programa. En *Obliq*, los objetos permanecen en el nodo en el que han sido creados, y los *closures* contienen referencias de red a estos objetos. Si se desea la migración del objeto es necesario programarlo explícitamente clonando objetos remotamente y borrando después los objetos originales.

El lenguaje Java fue diseñado específicamente para soportar el desarrollo y la distribución de aplicaciones a través de WWW. Sin embargo, con el paso del tiempo el lenguaje se está revelando como una atractiva, potente y potencial plataforma de sistemas ubicuos [5]. De este modo, Java se ha propuesto también como un lenguaje para el desarrollo de protocolos que mejoren el rendimiento y soporten nuevas aplicaciones en todos los entornos. No hay que perder de vista que el mayor inconveniente para el desarrollo de nuevos protocolos y servicios de comunicaciones lo encontramos en la necesidad de actualizar varios millones de sistemas finales y nodos de interconexión con implementaciones compatibles entre todos ellos. Para vencer esta preponderancia de los protocolos actuales lo que se viene haciendo es diseñar protocolos a medida, incorporando en las capas de aplicación código específico para labores muy concretas, lo que impide una vieja aspiración de la programación que es la reusabilidad.

Java aporta sobre todo una importante característica como es la de ser una plataforma que permite el desarrollo de aplicaciones portables e independiente de la arquitectura de los sistemas. Esto, gracias a la aportación de su *virtual machine* está haciendo que el lenguaje se convierta en una potente herramienta para el desarrollo y expansión de protocolos de comunicación y también para la implementación de SMA por su carácter inherentemente distribuido.

En la actualidad se está empleando Java combinado con WWW como una plataforma para el desarrollo de protocolos de comunicaciones. Aunque nuestro objetivo es ligeramente diferente a esta visión, podemos destacar las siguientes características que hacen de Java una atractiva plataforma para el desarrollo de un

prototipo del protocolo TAP:

- Los protocolos, y el código adicional que puedan necesitar para funcionar, puede ser descargado y ejecutado en la propia red cuando sea necesario. Esto permite la interoperatividad y flexibilidad entre los extremos de la comunicación y los propios nodos de interconexión (AcTMs en nuestro caso). Esta característica hace de Java una de las mejores herramientas para el diseño y desarrollo de los conceptos presentados en el *Capítulo 6* sobre los sistemas multiagente y las redes activas.
- Por otro lado, el problema de la portabilidad del código de los protocolos es reducido con el uso de Java tal como se demuestra en [6], gracias a la posibilidad de aprovechar la máquina virtual de Java que es la que realmente se porta en lugar de los protocolos.
- Las características de la orientación a objetos como es la herencia, la especialización y el polimorfismo aportan nuevas posibilidades a los programadores y facilitan la reusabilidad del código.
- No obstante, estas ventajas tienen en la actualidad un coste debido a que la máquina virtual es emulada en software y, mientras no se emplee soportada en hardware, esto implica una apreciable caída en el rendimiento de la red, lo que es importante en el caso del desarrollo de protocolos.

En la actualidad Java está formada por dos entidades separadas: un lenguaje de programación [7] y por debajo de éste una especificación de máquina llamada JVM (*Java Virtual Machine*) [8]. Estas dos partes juntas pueden entenderse análogamente a lo que podemos encontrarnos al combinar el lenguaje de programación C y la arquitectura Sparc. El lenguaje de programación es, por sí mismo, orientado a objetos, y más simple que otros como C++ y Smalltalk.

JVM interpreta los bytecodes Java que son instrucciones ensambladas por una arquitectura de CPU abstracta. Cuando el código escrito en Java es compilado en formato bytecode interpretado por la JVM, los programas Java podrán ser ejecutados en una amplia gama de arquitecturas de computadores sin necesidad de recompilación previa, lo que aporta esa característica tan importante de portabilidad de Java. La mayor ventaja del formato en código binario es que es independiente de las arquitecturas hardware, de los interfaces de sistemas operativos y de los entornos de GUI (ventanas). Por tanto, el compilador de Java no genera código máquina relativo al juego de instrucciones del hardware nativo, sino que genera bytecodes independiente de la máquina real, para una máquina hipotética que es implementado por el intérprete Java y el sistema *run-time*.

El lenguaje de programación Java dispone de una serie de librerías de clases predefinidas para la manipulación de interfaces gráficos, flujos básicos de E/S, utilidades como la gestión de tablas de hashing y vectores, y también relativos al networking. El soporte estándar, en cuanto al networking se refiere, está limitado a los servicios que usan los protocolos TCP y UDP y algunas otras funciones relacionadas. El lenguaje también aporta clases para la comunicación entre servidores de web y todos los aspectos relacionados con estas funciones.

Sin embargo, Java permite la extensión de nuevas clases que pueden ser cargadas dinámicamente en la JVM cuando son necesarias. Esto nos va a permitir la implementación de nuevas clases que constituyan los diferentes subsistemas del protocolo TAP. Como puede observarse en la *Figura 12.2*, los programas Java son aplicaciones que se ejecutan como intérpretes independientes y autónomos [1]. En la Figura puede observarse el cargador de clases, desde el que la JVM puede ofrecer la posibilidad de cargar y añadir dinámicamente nuevas clases en el sistema. El cargador de clases recibe las peticiones de nuevas clases y después las carga desde ficheros (si el intérprete que usa la clase se está ejecutando como una aplicación independiente) o desde la red. En nuestra propuesta de prototipo de TAP se emplea la capacidad para cargar y ejecutar protocolos (algoritmos) “en el aire”, o que son aportados por el cargador de clases.

En los protocolos desarrollados en lenguajes de programación tradicionales como C se emplean de forma habitual definiciones comunes a varios programas recurriendo a ficheros de cabecera (ficheros *.h*) y también constantes globales y preprocesado de símbolos. Pero Java no soporta ninguna de estas posibilidades ampliamente usadas en C, por lo que los ficheros cabecera con definiciones, constantes y máscaras de bits comunes no pueden usarse. Pero en la práctica este tipo de limitaciones no es ninguna restricción ya que el programador puede declarar constantes y definiciones como clases de variables en aquellas clases que sean compartidas. Por ejemplo, en el caso de TAP empleamos constantes como los VPI/VCI y también los nombres de los agentes en una clase, de forma que otros protocolos o algoritmos puedan referenciarlos.

[1] Si en la *Figura 12.2* se cambian las aplicaciones por *applets* y se añade un navegador de web entre el SO y la JVM tendremos aplicaciones Java ejecutándose restringidas a un intérprete que estará contenido en el navegador de web específico que esté instalado.

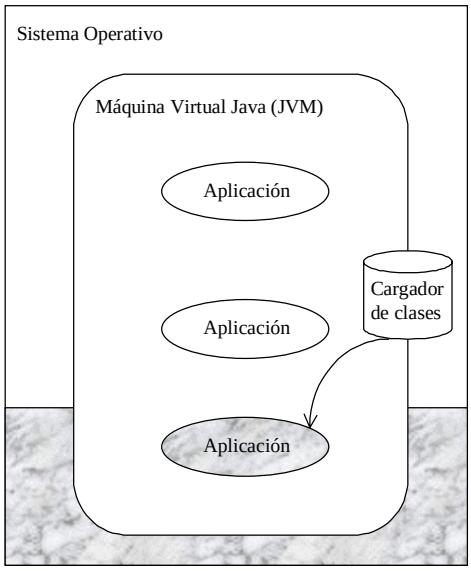

Figura 12.2. *Ejemplo del modelo de arquitectura de la JVM como sistema independiente*

Colocando definiciones comunes, máscaras de bits y constantes en una clase Java dispondremos de varias ventajas [9]. Primero, si nada cambia, los protocolos que ya hayan sido desarrollados en otros lenguajes de programación deberán ser completamente compilados y redistribuidos a los sistemas afectados. En los protocolos basados en Java, cambiando constantes, máscaras de bits o definiciones comunes sólo se vinculan cambiando la clase apropiada y descargando las versiones antiguas de todos los sistemas. Los protocolos que dependen de esta clase sólo necesitan actualizar el cambio la próxima vez que referencian a la clase. Segundo, los protocolos que emplean este planteamiento para compartir información de configuración, pueden ver modificado su comportamiento de forma más sencilla desde otros protocolos, algoritmos o desde las propias aplicaciones de los usuarios. Esta es precisamente una de las más atractivas características de Java para diseñar redes programables bajo las premisas de los agentes software o de las redes activas.

La implementación de la estructura de un protocolo identifica cómo éste interactúa con otros protocolos y con el resto del sistema. Por esto define los interfaces y métodos para construir la tabla del protocolo que comentamos anteriormente, y la forma en que se ejecuta el protocolo. Así, la estructura del protocolo definida para el uso con protocolos basados en Java es similar a la implementación tradicional donde se definían las entradas, salidas y los métodos de control. Sin embargo, como se destaca en [5], la estructura de protocolos basados en Java difiere de los métodos tradicionales en varios aspectos y en nuestro caso particular hemos de destacar los siguientes aspectos de interés que nos afectan más directamente:

- Los protocolos basados en Java se implementan como clases que son instanciadas por los usuarios o por otros protocolos que requieran de sus servicios.
- Las clases de los protocolos heredan su estructura de protocolo de una clase de protocolo base. El uso de la herencia para imponer la estructura del protocolo permite a éstos aprovechar el soporte del lenguaje para la programación estructurada, permitiendo al programador ampliar sus funcionalidades a través de la especialización, mientras se soporta la reusabilidad del código del protocolo.
- La estructura del protocolo puede definir métodos explícitos para la composición de protocolos en tablas. Por ejemplo, en el caso del SMA-TAP usamos estos métodos para identificar protocolos vecinos superiores e inferiores a un protocolo dado para identificar la jerarquía del SMA completo dentro de un AcTMs. Así, *protprev* y *protsig* identifican en la tabla de protocolos al protocolo vecino anterior y posterior respectivamente. En la estructura del protocolo pueden emplearse también métodos explícitos como *FinConex()* para cerrar una conexión concreta ya que Java no dispone de memoria explícita, ni permite el uso de punteros ni de desasignación de objetos.
- Por último, disponemos de la ventaja del uso del multithreading, de forma que se pueda emplear un hilo por objeto de protocolo. El soporte para la construcción de hilos es una de las herramientas más potentes de Java para la ejecución de eventos concurrentemente. El multithreading es la forma de conseguir rapidez y concurrencia de una forma sencilla con un solo espacio de proceso. Java aporta en su librería la clase *Thread* con un completo conjunto de métodos que permite que un thread sea: arrancado, parado, ejecutado, chequeado, etc. Además, puede hacerse uso de primitivas de

sincronización basadas en el paradigma monitor y condición variable. En el caso de TAP el uso de los hilos es una de sus principales características ya que, cada uno de los agentes del SMA dispone de hilos de comunicación con los agentes con los que interactúa, de forma que estos hilos se emplean para el intercambio de información de comunicación (conversación) entre los agentes, y otros para la transmisión de información de datos (células y PDU) que intercambian entre sí. Los hilos se emplean para el estudio del comportamiento periódico; para el procesamiento específico de cada protocolo como la creación y mantenimiento de las tablas de rutas en el establecimiento de la conexión; para la asignación de VPI/VCI; para el mantenimiento de las colas de entrada, de la DMTE, de las tablas de E/S y también para el mecanismo de solicitud de retransmisiones mediante las células BRM. Este tipo de procesamiento periódico del rendimiento del protocolo simplifica su código y evita el uso de los clásicos *timeouts* en el diseño de protocolos. Los hilos aportan un importante paradigma a la programación con Java porque cada hilo posee su propia pila de llamadas, por lo que pueden ser activados múltiples hilos con cada protocolo o algoritmo de forma simultánea sin interferir con ninguno de los otros. Cuando un paquete concreto no puede ser procesado (cola llena, tablas llenas, sincronización rota, etc.) el hilo puede mantener el paquete autosuspendiéndose (con el consiguiente efecto sobre el rendimiento final) hasta que el procesado del paquete pueda continuar en lugar de encolar el paquete, activar un *timer* y terminar. En resumen, el soporte de los threads es una de las herramientas más poderosas de Java, no sólo para aportar la interacción entre aplicaciones gráficas, sino por poder ejecutar concurrentemente múltiples eventos. El soporte de los threads de Java incluye un sofisticado conjunto de primitivas de sincronización basadas en un potente y ampliamente usado paradigma monitor y variable condición. Para que un programa software sea autónomo, éste debe ser un proceso separado o un thread. En una aplicación multiagente, un agente puede ser un thread de control separado y lo normal es que cada agente tenga su propio thread. Cuando usamos threading la ventaja clave es que podemos conseguir ejecuciones simultáneas. En realidad, el multithreading se usa por cada entidad funcional en un agente como son comunicaciones entrantes y salientes.

Para una aplicación multithreading como es un SMA, es necesaria la sincronización entre los threads de agentes diferentes. En Java se usan monitores para coordinar múltiples threads usando los métodos *wait()* y *notify()* disponibles en cada objeto Java. Un thread espera por la ocurrencia de alguna condición o evento, y notifica un *thread wait* cuando se produce una condición o evento. Los threads son coordinados usando el conocido concepto de variable condición que es un estado lógico que debe mantenerse a *cierto* para que un thread pueda comenzar a actuar. Si la condición no se mantiene a *cierto*, el thread debe esperar hasta que la condición se convierta en *cierta* antes de continuar. En Java este comportamiento podría expresarse con el código presentado a continuación y usado en el agente RCA para que éste se mantenga a la espera mientras no se produzca una notificación de congestión en el buffer.

```
While (NoNotificacionCongestion) {
        wait();
}
```

Los métodos *wait()* y *notify()* deben ser invocados desde un método sincronizado o desde un estado sincronizado. Estos dos métodos simplifican la tarea de coordinación de múltiples threads en un programa concurrente escrito en Java. El siguiente fragmento de código presenta un ejemplo de cómo puede ser usada la variable de control en el agente DPA para esperar por una decisión hecha sobre el estado de la DMTE donde se invoca a una llamada a método remoto el cual llama al método *EstadoDecision()*. De este modo, DPA espera mientras no se reciba una notificación de retransmisión desde la DMTE.

```
Public sincro void SincEstadoDecision(int estado)  {
        EstadoDecision = estado;
        notify();
}
sincro(preparado) {
        if (EstadoDecision = = NoPreparado)
        wait();
        }
}
```

- Otra atractiva ventaja del uso de Java para la programación de agentes y protocolos es su posibilidad de emplear RMI (Remote Method Invocation) para crear aplicaciones Java-to-Java distribuidas, en las cuales los métodos de los objetos Java remotos puedan ser invocados desde otra máquina virtual que esté situada en otro nodo. En nuestro caso no hacemos uso de RMI, ya que el TAPS se ejecuta en su actual versión únicamente en un nodo, pero ésta es una de nuestras investigaciones futuras en un intento por conseguir un escenario real y plenamente distribuido del protocolo.

- El carácter orientado a objetos es otra de las características de Java para obtener código reusable y para encapsular todos los datos en objetos. Esta es una ventaja que empleamos ampliamente en el simulador con múltiples objetos que se van a usar en las transferencias y conversaciones.

- Además, Java mediante métodos nativos permite llamadas a código ya compilado de otros lenguajes (código nativo) optimizando la velocidad del código que implementa un agente concreto. El interface Java Native Method Interface permite a los programas que ejecutan la Java Virtual Machine (JVM) llamar a programas de interfaz escritos en C y C++. Esto permite que los programas escritos en otros lenguajes puedan arrancar la JVM y ejecutar programas Java.

- Los protocolos basados en Java pueden elegir si las entidades como mensajes, capas, conexiones, buffers, colas, tablas y tiempos son limitados a la capa de planificación. En bastantes casos se han implementado los paquetes acompañados a través de los protocolos mediante hilos no apropiativos, lo que quiere decir que cada protocolo dispone de su propio hilo para el envío de los flujos de datos hasta el siguiente protocolo o la apropiada estructura de datos. Es decir, cuando una aplicación, o uno de los algoritmos del conjunto de protocolos, necesita enviar datos, lo que se hace es crear un buffer al que se asigna un hilo de comunicación para acompañar los datos hasta la tabla de protocolos, de forma similar a como se hacía tradicionalmente con otros lenguajes de programación.

- Tanto para la implementación de protocolos como de SMA, Java aporta otra ventaja adicional que es la de poder usar una clase pensada para los sockets de datagramas multicast (*MulticastSocket*) que permite la formación de grupos multicast. Del mismo modo se puede emplear también para el envío de mensajes de difusión broadcast que pueden ser muy útiles para la difusión de acciones concretas a un grupo determinado de agentes. En la versión actual del SMA-TAP no se aprovecha esta posibilidad, pero es otra de las mejoras prevista para futuras versiones.

- La seguridad es otra de las cuestiones en el ámbito de los SMA y de los protocolos de comunicación, y en este sentido Java ha sido mejorado para aportar fiabilidad a las comunicaciones, incluso a través de técnicas de cifrado.

En las arquitecturas de protocolos basadas en Java, suelen usarse clases *servicio* especiales que construyen dinámicamente las tablas de protocolos en tiempo de ejecución cuando las aplicaciones necesitan servicios concretos de comunicación. Por ejemplo, si una aplicación necesita el servicio GoS únicamente tendrá que crear una instancia de su correspondiente clase *servicio*. El código de la clase servicio se encarga de asignar e inicializar las capas de protocolos e interconectarlas a través de los métodos ya comentados *protprev* y *protsig*. De este modo la clase *servicio* puede entenderse como un *daemon* del estilo de Unix que se encarga de la construcción y mantenimiento de una tabla particular de protocolo [6].

El cargador de clases de Java se emplea habitualmente en la construcción de la tabla de protocolos porque ofrece la posibilidad de incorporar nuevas clases de protocolos (tal como se hace con las clases ordinarias de Java) en la JVM. Las clases *servicio* pueden referenciar explícitamente a una clase de un protocolo concreto referenciando directamente su nombre. Si la clase no está ya cargada en la JVM, el cargador de clases del sistema lo hará de forma transparente al usuario y a los protocolos (cargando la clase directamente de un fichero) sin la intervención de la propia clase *servicio*.

Las clases *servicio* y otros protocolos basados en Java pueden cargar también explícitamente clases Java ellas mismas. Por ejemplo, envían a las otras clases código de clase Java como datos transmitidos vía conexión de transporte normal. Estos datos pueden ser convertidos en una clase Java ejecutable usando una de las facilidades de construcción del cargador de clases. Esta característica extensible de los protocolos con Java permite la introducción o reemplazamiento “en el aire” de nuevo código de protocolo, lo que podemos emplear para la implementación de los agentes programables en el caso de TAP.

Por otro lado, la tabla de cada protocolo puede ser colocada como una clase de información del protocolo o dentro de la información de estado de cada instanciación individual de esa clase. Si se hace en la clase fija esta tabla de configuración de protocolo se usa para cada instanciación de objetos del protocolo. Si se sitúa la tabla de configuración del protocolo en el objeto permite a cada sesión individual determinar y controlar su propia tabla de protocolo. En el caso de TAP, y con la intención de que cada algoritmo y conexión pueda

elegir los servicios que requiere de una forma flexible en sus comunicaciones, hemos decidido situar la configuración de los protocolos en la información de estado de cada sesión de los objetos, en lugar de hacerlo en una clase propia para ello. Esto lo hacemos a través de las operaciones de sincronización especificadas en los algoritmos que presentamos en el *Capítulo 11*. De este modo, se permite la existencia de múltiples tablas de protocolos conteniendo los mismos protocolos pertenecientes a agentes del SMA.

En lo referente al rendimiento de los protocolos implementados en Java cabe destacar que el hecho que la JVM sea implementada en software y que el propio lenguaje sea orientado a objeto, hace pensar que el rendimiento de los protocolos desarrollados en Java y compilados en bytecode Java pague un importante precio en cuanto a su rendimiento para disponer de la portabilidad y extensibilidad del lenguaje. Estos costes han sido evaluados en [6] comparativamente con lenguajes más tradicionales como C y Streams, en función de varios aspectos como el tiempo de procesamiento de los paquetes, el overhead debido a la recolección de restos de memoria libre (*garbage collector*), y a las técnicas de compilación no demasiado depuradas. Cabe destacar que la gestión de memoria en los programas escritos en Java es bastante más simple que otros lenguajes de programación, porque la memoria no necesita ser desasignada explícitamente debido a que el recolector de *garbage* de la máquina virtual se encarga de reclamar la memoria libre. Esta forma de actuar permite simplificar el código programado por poder obviar las operaciones de asignación y liberación de memoria típica en otros lenguajes; sin embargo, acaba pagándose también un precio en rendimiento por esta simplificación para el programador, ya que la máquina virtual debe suspender periódicamente la ejecución de los programas para localizar la memoria libre no utilizada. Esta forma de actuar acaba afectando por tanto al rendimiento de los protocolos y, sobre todo, a los que deben soportar servicios en tiempo real, ya que el recolector de *garbage* es invocado sólo cuando la máquina virtual está baja de memoria, y esta condición ocurre, desgraciadamente, de una forma no determinística.

Por tanto, la mayor parte de estudios de rendimiento de Java se centran en el intérprete de la JVM y en el propio lenguaje Java. No obstante, este tipo de estudios es realmente difícil de realizar debido a que Java se encuentra aún en su primera etapa como lenguaje, comparado con otros, y también debido al comportamiento de las diferentes plataformas sobre las que se puede ejecutar. Los resultados mostrados en [6] indican que la latencia introducida por Java en el procesamiento de un paquete con respecto a otros lenguajes nativos como C/BSD, Streams y x-Kernel (basados en código nativo compilado y ejecutable directamente), incrementa en un factor de 10 en el envío, y de 17 en la recepción del paquete. El resultado del estudio de rendimiento debe mejorar a medida que evolucione el propio lenguaje Java y la tecnología de su intérprete en la JVM (basada en técnicas de compilación *just-in-time*), y de hecho los estudios realizados demuestran que el rendimiento de JDK v. 1.1.1 ha doblado el rendimiento respecto a su versión previa. En realidad, la gran desventaja en cuanto a rendimiento está en la diferencia entre el código Java interpretado y el código nativo de otros lenguajes que es compilado y ejecutable directamente. De hecho, los estudios de rendimiento del uso de Java en el desarrollo de protocolos demuestran que, sobre una misma plataforma hardware, la compilación JIT (*Just-In-Time*) aporta mejoras sobre versiones como JDK (Java Development Kit), ya que JIT realiza compilación en tiempo de ejecución del bytecode de Java que es directamente usable por el sistema operativo que está debajo, y esto permite reducir parte de los problemas de rendimiento.

Por todas estas razones generales y por otras más particulares que podrían ser entendidas más puramente dentro del contexto del desarrollo de protocolos y de sistemas multiagente, se decidió adoptar el lenguaje Java para la implementación del simulador de protocolo TAP y, por tanto, el sistema multiagente que propuesto en esta tesis. Aunque Java es una de las posibles opciones, en la actualidad es el líder de los estándares de facto y el que han usado gran parte de los fabricantes de *networking* como Cisco ó 3Com, aunque tenga algunos aspectos mejorables como la portabilidad, seguridad y sus características de rendimiento en comunicaciones. Sin embargo, el lenguaje Java se ha demostrado que puede aportar atractivas ventajas para la construcción de sistemas multiagente complejos.

12.3. METODOLOGÍA Y DECISIONES DE DISEÑO DE TAPS

Hemos podido ver ya las ventajas más importantes del uso del lenguaje Java para el desarrollo, tanto de protocolos, como de SMA. Con este punto de partida, en este apartado se discute la metodología y las decisiones de diseño seguidas para el desarrollo del SMA-TAP que es la principal parte del TAPS. Así debemos de conseguir la comunicación, coordinación y cooperación entre los agentes para lograr una solución coherente.

Precisamente uno de los principios de diseño deseados es el de huir de la excesiva centralización y, tal como hemos podido comprobar en el capítulo anterior y en los precedentes, en el caso de TAP no se dispone de un control central en el sistema, aunque partimos de una tecnología inherentemente orientada a la conexión, por lo que se depende de ciertos aspectos como es el control de los extremos de la comunicación

con algunas características que han de mantenerse en toda la conexión. Este objetivo de conseguir un sistema lo más distribuido posible nos ha llevado a emplear los métodos de comunicación directa entre los agentes. El método de federación es más apropiado para SMA con un gran número de agentes y el método *blackboard* es claramente centralizado. Con el modo de comunicación directa aportamos a TAP las posibilidades básicas en todo SMA. Como se ilustra en los algoritmos del *Capítulo 11*, en algunos casos se necesita comunicar con varios agentes a la vez por lo que el modo de comunicación por difusión es el adecuado, pero esta posibilidad la resolvemos con comunicación directa ya que en todos los casos se trata de comunicación entre pares de agentes por lo que el método directo es el más apropiado.

La metodología de diseño del SMA-TAP que hemos empleado está basada en [1,10] y se divide en las siguientes etapas:

- Identificación de los agentes: En esta etapa realizamos el análisis del sistema y de los objetivos que perseguimos y que, en esencia, ha sido ya descrito en los *Capítulos 6*, *10* y *11* donde hemos identificado todos los agentes que intervienen en el sistema. Tenemos también ya una lista de las funciones particulares que desempeñan cada uno de los agentes y del objetivo general del sistema completo. De este modo tenemos definido un conjunto de entidades que interactúan unas con otras para lograr ese objetivo general, tal como se ha descrito en el *Capítulo 10*. Estas entidades identificadas son las que van a constituir los agentes del sistema, en los que se han aclarado los objetivos de cada agente y también los servicios que aportan al SMA. Además, hemos dividido ya esos agentes en grupos según sus categorías o clases como agentes programables (CoSA y CCA), agentes autónomos (DPA) y colaborativos (CoSA, RCA y WFQA).

- Identificación de las conversaciones entre agentes. El problema de la coordinación puede ser entendido conociendo los procesos de interacción entre los agentes del SMA. Pero este conocimiento ha de ser sobre la capacidad para resolver el problema completo desde el punto de vista de todo el SMA y no desde el de un agente concreto. Puede verse el sistema como un conjunto de elementos, de modo que la interacción entre esos elementos puede entenderse como un modo de conversación. O más concretamente, una conversación puede entenderse como la planificación de un agente para conseguir un objetivo, basándose para ello en la interacción con otro agente [1]. Este tipo de conversaciones suele modelarse mediante autómatas y, en nuestro caso, hemos decidido explicarlo sin excesivos formalismos en el *Capitulo 10*, donde vieron las posibles conversaciones que cada uno de los agentes puede emprender. Así se tienen las diferentes clases de conversaciones del sistema y las clases de aplicaciones específicas que puede usar cada conversación.

- Identificación de las reglas de conversación, que permite fijar las normas de establecimiento de conversaciones entre agentes; es decir, una determinada acción produce un estado en la conversación, y el estado actual de la conversación puede influir en el siguiente estado del agente. Esta etapa puede formalizarse también mediante autómatas, aunque nosotros hemos especificado este tipo de precedencias entre estados y conversaciones de los agentes literalmente en el *Capítulo 10* y en forma de algoritmo en el *Capítulo 11*.

- Análisis del modelo de conversación ya especificado en la etapa 2, que intenta encontrar incoherencias en el sistema. Analizando la coherencia de cada agente se llega a garantizarla en todo el sistema. Cuando se detectan inconsistencias en el sistema se vuelve a la etapa 2 para redefinir las conversaciones. Para todo este análisis suelen emplearse máquinas de estados finitos y redes de Petri y, en nuestro caso, hemos podido comprobar este análisis mediante trazas del funcionamiento del sistema que han sido presentadas en el *Capítulo 10*.

- Implementación del SMA que es el último punto de esta metodología, donde debe elegirse un lenguaje o una herramienta de implementación del SMA que sea capaz de asegurar la comunicación, interoperación y coordinación entre todas las entidades del sistema. Los lenguajes (o herramientas de elaboración de SMA) deben permitir soportar el tipo de la comunicación deseada (directa, broadcast, federación o *blackboard*), la creación de agentes y de sus hilos, los mensajes de comunicación, la iniciación de la conversación de cada agente en su hilo y el desarrollo de reglas de conversación. Los lenguajes como Java pueden verse como este tipo de herramientas que aportan las características que hemos podido comprobar en el apartado anterior para la implementación de los SMA. No obstante pueden emplearse también herramientas para el desarrollo más sencillo de SMA como JAFMAS [1], JATLite[10] y otras tantas que suelen aportar un conjunto de clases Java ya implementadas que pueden ser usadas para la implementación de SMA.

La capacidad de comunicación es la que permite a los agentes de un SMA coordinar sus acciones y cooperar con otros agentes. Generalmente la comunicación puede establecerse para el intercambio de

mensajes de forma directa, o bien a través de un sistema federado, mediante mensajes tipo broadcast o a través de un mecanismo de almacenamiento de datos compartidos donde se almacenan y recuperan los mensajes. Como hemos indicado antes, en nuestro caso implementamos el SMA basado en un sistema de comunicación directa sin tener en consideración los aspectos de enlaces físicos directos ya que el SMA se implementa en un solo ordenador en el que se simula el comportamiento de TAP. No obstante, en el mecanismo de inicialización de la comunicación podemos considerar que empleamos el sistema de comunicación de mensajes broadcast ya que, como se vió en el *Capítulo 11* (*Figura 11.1*), al ejecutarse el algoritmo TAP en el terminal emisor se realiza la inicialización de todo el SMA mediante el establecimiento de un mecanismo de sincronización con cada uno de los agentes que intervienen en la comunicación. El mecanismo de comunicación empleado a menudo depende de las dependencias del sistema y de las circunstancias en las cuales los agentes intercambian información. Existen tres mecanismos básicos de comunicación: llamada a procedimiento, *callback* y *mailbox*. En TAP se emplean llamadas a procedimiento.

Puede usarse también un tipo de mecanismo *mailbox*, un modelo cola y servidor. En un entorno de red con sus retardos inherentes, los agentes no pueden ofrecer esperas en torno a algo que vayan a necesitar en un punto futuro. De esta forma, los mensajes entrantes son colocados en las correspondientes colas. El servidor chequea cada mensaje de la cola por turnos o acordando una prioridad predefinida, por ejemplo, manipulando solicitando manipulación de conexiones antes de la configuración de la conexión. Este es el tipo empleado por el agente WFQA para recibir las notificaciones de retransmisión que le llegan desde el agente RCA.

La cooperación entre agentes permite crear una comunidad de agentes especializados que aúnan sus recursos y capacidades con la intención de solventar problemas complejos, pero con el coste adicional del overhead en los aspectos de comunicación. La comunicación activa los agentes en un sistema multiagente para intercambiar información bajo las bases de que cada agente coordina sus acciones y coopera con otros. Cuando los agentes necesitan conversar con otro agente, pueden hacerlo de varias formas. Pueden hablar directamente con otro agente, hablando el mismo lenguaje. O pueden hablar a través de un facilitador, ofreciendo una forma para comunicarse con él, y éste puede hablar con el otro agente. El SMA no emplea facilitadores, ya que cada par de agentes conversa de forma directa.

La interacción es otra implementación importante en un sistema multiagente, como el proceso de interacción que permite a varios agentes combinar sus recursos y acciones para alcanzar sus objetivos basándose en un concepto puramente distribuido. Sin embargo, la naturaleza distribuida y heterogénea de los sistemas multiagente hace que la implementación de la interacción entre agentes sea realmente compleja. Precisamente, trabajando en un SMA de agentes distribuidos lo más complejo es diseñar un mecanismo de comunicación o lenguaje que permita a un agente comunicar con los demás intercambiando mensajes y esto [1] es lo que diferencia precisamente a los agentes de los objetos.

El comportamiento de un agente puede caracterizarse como una máquina de estados finitos, la cual describe la interacción desde el punto de vista de un agente individual con todas sus posibles interacciones, en las cuales el agente puede establecer contactos con otros agentes. Sin embargo, los agentes son desconectados de otros teniendo un thread especial y poniendo las solicitudes en la correspondiente cola de mensajes. Cada cola puede intercambiar mensajes con conversaciones estructuradas con otros agentes, cambia de estado y realiza acciones.

La coherencia y la coordinación son dos más de los aspectos básicos en los SMA, mediante los cuales se establece en el caso de la coherencia el comportamiento completo del sistema al resolver problemas concretos para los que ha sido diseñado; y en de la coordinación la posibilidad de que un conjunto de agentes puedan interactuar para realizar acciones colectivas como las que se realizan entre RCA, CCA y WFQA para ser capaces de recuperar las PDU que se pierden en las congestiones.

12.4. ESPECIFICACIÓN DE LAS CLASES DEL SIMULADOR TAPS

El punto anterior ha presentado el diseño del SMA-TAP. El simulador de TAP (TAPS) se ha desarrollado en JDK (Java Development Kit) 1.1. En este apartado vamos a especificar (obviando los procesos de formalización de todos estos procesos) y centrándonos principalmente en identificar las clases propuestas y sus procesos de comunicación y coordinación y obviando los aspectos lingüísticos y semánticos.

Las clases de TAPS soportan agentes con múltiples hilos (uno para el propio agente, y uno por cada una de las conexiones que mantiene el agente con otras entidades).

A continuación se presenta una relación de las clases de mayor interés empleadas para la codificación del simulador. Estas clases son añadidas al cargador de clases de la JVM de forma que podrán ser empleadas por cada una de las entidades que necesiten usarlas para su ejecución. Se han reinterpretado algunas de las clases

propuestas en [1] y [10] y, a la vez, hemos incluido en el diseño nuestras propias clases que conforman la mayor parte del SMA. Se destaca en algunas de las clases propuestas sus atributos y métodos para clarificar su instanciación.

- *TAP*: es la clase principal que llama a los procedimientos de inicialización del resto de las clases que se presentan a continuación.
- *CrearAgente:* Esta clase permite la creación de los agentes del sistema en el proceso de inicialización del SMA-TAP. Es decir, cuando se arranca el protocolo TAP, la primera operación que realiza es la creación de los agentes del SMA. Para ello esta clase dispone del método *CrearAgente(),* que emplea el nombre de cada agente y sus atributos y, a continuación, se crea y se inicia (mediante el método *inicia*) arrancando sus correspondientes hilos de comunicación.
- *Mensajes:* Los agentes del SMA emplean esta clase para su comunicación entre ellos. Pueden observarse los atributos y métodos de la clase en la *Figura 12.3*, donde puede verse una variable *acción* que es usada en cada instanciación de la clase. Esta variable se usa para definir la acción interactiva que se va a establecer como *aceptar* y *rechazar*. Los métodos *SincEmisor()* y *SincReceptor()* se emplean para especificar los emisores y receptores de los mensajes. Por otro lado, el método *SincIntencion(),* se emplea para que el emisor pueda notificar la intención o el propósito de envío de mensaje, y el agente receptor para poder recibir esta intención. Pueden enviarse y recibirse mensajes de dos tipos: de *sincronización* y de *contenido*. Los de *sincronización* se emplean para establecer y mantener las comunicaciones entre los agentes con las acciones y estado de éstas. Los mensajes de *contenido* donde se indica la información que se envía a través de los mensajes. La variable *contenido* permite a los agentes especificar el contenido de sus mensajes, de forma que un mensaje *contenido* llevará la información que un agente va a enviar y el tipo de objeto a que pertenece y, para ello, emplea el método de sincronización *SincContenido().*

Clase Mensaje	
Variables	**Métodos**
accion *emisor* *receptor* *contenido*	*Mensaje(accion)* *Mensaje(mensaje_recibido)* *SincReceptor* *SincContenido* *SincIntencion* *TomarAccion* *TomarEmisor* *TomarReceptor* *TomarIntencion* *TomarContenido*

Figura 12.3. *Clase Mensaje del SMA-TAP*

- *ColaMensaj:* El SMA-TAP emplea esta clase para mantener una cola de mensajes cruzados entre los agentes. Esta clase dispone de métodos para mantener la cola y para realizar búsquedas: *AñadMensaj(), BorrMensaj(), BuscPorEmisor(), EsVacia(), Tamaño(),* etc.
- *PDU*: es una clases genérica de la cual heredarán todos los tipos de paquetes empleados en la aplicación. Una PDU es una estructura del tipo matriz en la que se van insertando datos de tipo entero, short, string, etc. Entre otros cuenta con el método *siguientePDUid().*
- *CélulaATM*: clase que cuelga de PDU, con las funciones y campos de las células estándares ATM.
- *General*: es una clase creada para contener todas las constantes de la aplicación y todos los procedimientos útiles, como la conversión de datos.
- *EAAL5***:** Clase que representa la capa EAAL-5 en los nodos emisor y receptor.
- *ATM:* Clase que representa la capa ATM en los nodos emisor y receptor.
- *Fisica:* Clase que representa la capa Física en los nodos emisor y receptor.
- *Fichero*: emplea la clase *File* de Java, y añade funciones específicas de la aplicación para leer y escribir ficheros de texto (leer línea, leer cadena, leer número, etc.).

- *Ventana*: representa el contenido de la ventana de datos genéricos.
- *VentanaAcTMs*: representa el contenido de la ventana de datos concretos del nodo activo.
- *Config*: clase para la ventana de edición de los ficheros de configuración.
- *Comunicacion*: Como se ha comentado, el SMA emplea el método de comunicación directa para lo que se dispone de esta clase que se emplea para enviar y recibir mensajes, tanto en los procesos de inicialización del SMA cuando se establece una comunicación con GoS, como entre los propios agentes que forman parte del SMA-TAP. Cada uno de los agentes dispone de una instanciación de esta clase que realiza la función de enviar y recibir mensajes directos. Para ello la clase dispone del método *SincMensajRcbd()* para recibir mensajes de otros agentes que, además, es usado por los agentes del SMA para enviar mensajes al agente al que pertenece el objeto asociado *Comunicacion*. Esta clase dispone también del método *EnvMensaj()* para enviar mensajes directos desde el agente. Los agentes envían mensajes, instanciando la clase *Mensaje*, como un argumento del método *EnvMensaj()*. Asociado a esta clase existe un hilo de comunicación, extendiendo la clase *Thread* incluida en el lenguaje Java de forma que cada objeto *Comunicacion* dispone de un hilo de ejecución separada, de modo que los agentes tendrán un objeto hilo *Comunicacion* separado.
- *ReglasConv*: Es la clase que se emplea para establecer las reglas de conversación entre los agentes que describen las acciones a realizar cuando una conversación entre agentes se encuentra en un estado concreto. Cada regla tendrá un estado en cada momento y un estado siguiente al que se llega en la ejecución. Esta clase se encarga de activar el mensaje recibido y el que va a ser transmitido a través de los métodos *SincRcbdMensaj()* y *SincTrmtdMensaj()*. El método *ControlRegla()* se ejecuta continuamente sobre la lista de reglas a través de un atributo *regla* y, cuando coincide con el atributo del estado actual de una regla, se usa el método *Invocar()* de la clase para ejecutar esa regla. Cuando esto ocurre se emplea el método *Transmitir()* para enviar un mensaje, que puede ir precedido y/o seguido de otras acciones (a través de los métodos *Antesde()* y *Despuésde()*) en función del mensaje de que se trate. Por ejemplo, si estamos ante el mensaje de notificación desde RCA hasta WFQA de una petición de retransmisión, se esperará una confirmación desde éste para generar una célula RM.
- *Gráfico*: representa a la ventana de simulación de la red. Esta clase usa los objetos gráficos que representan a los nodos y a sus enlaces. Esta clase incluye además la siguiente clase:
  - ✓ *DibujoNodo*: incluye todos los elementos necesarios para representar a un nodo de la red con sus propios enlaces.
- *NodoAcTMs*: es la clase más amplia, ya que representa a un nodo activo de la VPN. Contiene las variables locales y los procedimientos necesarios para simular el comportamiento de un nodo activo que implementa TAP. Dentro de esta clase existen las siguientes clases:
  - ✓ *TablaRouting*: representa la tabla de routing del nodo activo empleada para mantener los datos de cada conexión identificada por un VPI/VCI. Esta tabla dispone de métodos para la consulta y el mantenimiento de la tabla.
  - ✓ *MonitorBuffer*: clase empleada en todas las entidades que envían o reciben datos. Representa la estructura de datos buffer de los conmutadores activos. Los buffers almacenan PDU y células ATM genéricas de forma que pueden instanciarse para tratar, tanto células de fuentes no privilegiadas, como PDU pertenecientes a conexiones garantizadas.
  - ✓ *MonitorColas*: clase empleada también por todas las entidades que envían o reciben datos. Representan las estructuras de datos colas de entrada del tráfico a los conmutadores. Pueden almacenar, tanto PDU, como células ATM de forma que puedan instanciarse para tratar, tanto datos en forma de PDU, como en forma de células.
  - ✓ *MonitorDMTE*: clase creada para uso de las entidades que acceden a la memoria dinámica, tanto para leer, como para copiar unidades de PDU. Se emplea por tanto para representar la estructura de datos DMTE en la que se almacenan las PDU de las conexiones con GoS.
  - ✓ *MonitorTablaE/S*: clase pensada para la estructura de datos de las tablas de E/S asociadas a los puertos de salida. Esta clase es también empleada por todas las entidades que envían o reciben datos y representan los enlaces entre unas entidades y otras para el flujo de la información.
  - ✓ *AgenteCoSA***:** Clase que representa al agente de Clase de Servicio.
  - ✓ *AgenteWFQA***:** Clase que representa al agente que aplica el Weighted Fair Queueing.

- ✓ *AgenteCCA*: Clase que representa al agente de Control de Congestión.
- ✓ *AgenteRCA*: Clase que representa al agente de Control de Retransmisiones.
- ✓ *AgenteDPA*: Clase que representa al agente Despachador de las PDU a los puertos de salida.
- ✓ *CélulaBRM*: clase que cuelga de PDU, con los campos específicos de las células backward resource management, que empleamos para la solicitud de retransmisiones.

- *NodoATM*: Clase que representa los nodos estándares ATM que no implementan TAP por lo que no se les considera nodos activos. Como los nodos no activos sólo reenvían el tráfico de datos hacia el siguiente nodo en dirección al destino, no necesitan las estructuras de datos de la clase *NodoAcTMs* ni tampoco las clases de los agentes software. Como se encargan de la conmutación de las células sí disponen de la clase *TablaRouting*. Además, como estos conmutadores reenvían las células BRM que reciben en dirección al nodo emisor también emplean la clase *CelulaBRM* de la clase *NodoAcTMs*.

Cada una de estas clases se implementa como un módulo independiente que contiene la descripción de los procedimientos que componen cada clase. En los casos en que una clase incluya a su vez otras clases, cada una de estas dispone a su vez de sus propios procedimientos que implementan la clase de que se trate.

Un SMA se compone de diferentes clases de agentes, de forma que una vez que se han especificado los aspectos relativos a la comunicación, coordinación y cooperación entre los agentes, lo que queda es disponer de los agentes del SMA. Se ha visto ya que disponemos de clases para la creación de los agentes en el SMA-TAP y también clases específicas para cada uno de los agentes. A continuación realizaremos la especificación de una clase de agente general que permita, mediante instancias, el desarrollo de los diversos agentes del sistema y a la vez nos sirva para explicar la forma en que operan, así como conocer su arquitectura interna y los métodos y atributos que lo forman.

Cada uno de los agentes estará compuesto de las clases relativas a los aspectos de comunicación así como a los de modelo social (coordinación y cooperación). La *Figura 12.4* presenta el aspecto interno de este agente general en el que pueden observarse las clases comentadas interactuando entre sí para conseguir que un agente pueda comunicarse, coordinarse y cooperar con otros agentes del sistema.

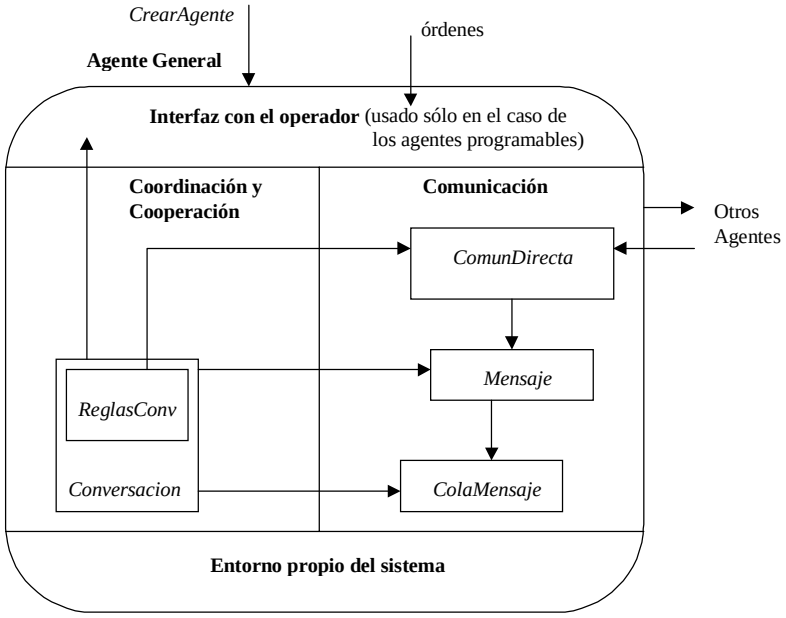

Figura 12.4. *Arquitectura interna de un agente general*

A continuación se presenta la clase *Agente* desde el punto de vista general sin centrarnos en ninguno de los cinco agentes que constituyen el SMA-TAP, pero abarcando todos sus métodos, y sabiendo que los cambios más importantes con que podemos encontrarnos tienen que ver con los agentes con los que se comunica cada agente particular. Así, en la *Figura 12.5* se presentan los atributos y métodos de esta clase *Agente*, que extiende la clase *Threads* de Java, ya que cada agente en el lenguaje Java dispone de su propio hilo de ejecución, lo que permite que varios agentes diferentes puedan ejecutarse concurrentemente aunque hayan sido creados con el mismo espacio de código de programa.

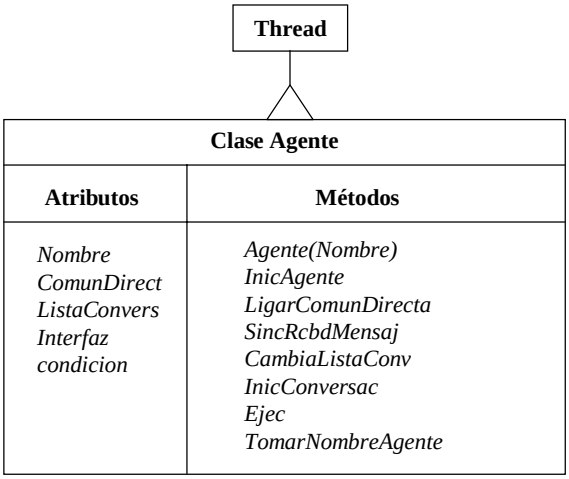

Figura 12.5. *Métodos y atributos de la clase Agente*

Una vez creado, un agente mantendrá su sincronización con los agentes con los que se comunica, por lo que podrá recibir y enviar mensajes a éstos. Pero antes de esto deberá registrar su(s) agentes de comunicación directa para lo que emplea el método *LigarComunDirecta(),* y a continuación se instancia el objeto *ReglaConv* a través del método *InicAgente()* de la clase *Agente*. Cuando esto ha concluido es cuando se inicia la ejecución del agente con el método *Ejec()* y cuando el agente estará listo para enviar y recibir todo tipo de mensajes. Mediante el método *CambListaConv()* se actualiza la lista de conversaciones que cada agente mantiene. Además, se define el método *InicConversac()* para determinar las condiciones bajo las cuáles cada agente va a comenzar una conversación con otro agente (por ejemplo WFQA acepta desde CoSA cuando dispone de colas vacías). La *Figura 12.6* (adaptada de [1]) presenta la secuencia de pasos por los que discurre el ciclo de vida, en este caso del agente CoSA, para describir el funcionamiento de este agente particular aplicando los conceptos comentados en el caso general.

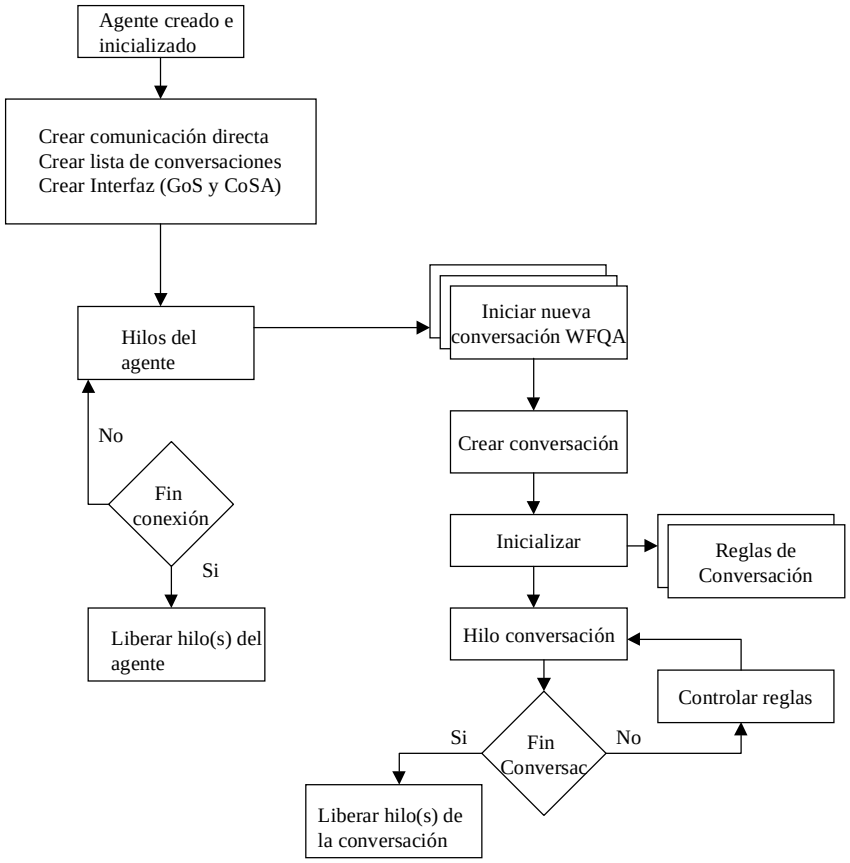

Figura 12.6. *Ciclo de vida de las operaciones del agente CoSA*

12.5. EVALUACIÓN DE RESULTADOS

En los trabajos [11,12] presentamos el buen comportamiento del protocolo RAP (la versión previa del protocolo actual) sobre la arquitectura TAP. Posteriormente implementamos y simulamos la actual versión [13,14] del protocolo confiable incluyendo las técnicas software ya explicadas para introducir las características activas en los conmutadores. Estos mecanismos controlan y gestionan los VCI privilegiados y también ofrecen el mecanismo de recuperación de las PDU congestionadas de los conmutadores activos vecinos.

Como se ha comentado en puntos anteriores se ha implementado en Java un simulador como prototipo de TAP y para evaluar el comportamiento del protocolo. El simulador permite definir la probabilidad de congestión en los emisores, receptores y en los conmutadores ATM. La simulación también permite al usuario introducir valores variables como los parámetros de las fuentes ON/OFF (T_{on} y T_{off}), el número de fuentes emisoras y de receptores del tráfico o el número de conmutadores activos y no activos. El simulador permite al operador de la red la gestión de los agentes programables CoSA y CCA.

Para analizar el comportamiento de la pérdida de células en las simulaciones se usan fuentes de tráfico ON/OFF ya que permiten modelar el tráfico a ráfagas característico del tráfico UBR o ABR para el que TAP es más apropiado. Además, el modelo ON/OFF suele usarse para caracterizar el tráfico ATM por conexiones unidireccionales.

La fuente genera ranuras de tiempo vacías. Se usa en todos los ejemplos una *CSR (Cell Slot Rate*) de 353.208 células/s sobre un modelo de red con enlaces de 155,52 Mbit/s. Cuando la velocidad de llegada de células CAR es menor que CSR, habrá ranuras de tiempo vacías durante el estado activo (ver *Figura 11.12*).

Como se comentó en el *Capítulo 11*, en las simulaciones se emplean valores estadísticos para los tiempos de actividad y de silencio de las fuentes (T_{on} = *0,96* s. y T_{off} = *1,69* s.), aunque también hemos usado otros valores para analizar sus efectos sobre TAP. Se destaca que se han variado algunos de los parámetros para analizar el comportamiento de TAP cuando varía el escenario y los descriptores de las fuentes de tráfico, según se estudia en esta sección.

La *Tabla 12.1* muestra lo descriptores de tráfico con sus valores máximos y mínimos usados en las simulaciones. Utilizamos un proceso que conmuta entre un estado de silencio (*idle*), y el estado activo (*sojourn*) que produce una velocidad media de células (entre 64 Kbits/s y 25 Mbits/s) agrupadas en PDU de 1.500 octetos, aunque el tamaño de las PDU es también un parámetro variable en las simulaciones. Durante los estados ON estos procesos generan células a una velocidad de llegada CAR.

TABLA 12.1
DESCRIPTORES DE FUENTES DE TRÁFICO ON/OFF

Descriptores de tráfico	Parámetros	Mínimo	Máximo
Velocidad Media de las Fuentes	*VMF*	64 kbit./s.	25 Mbit./s.
Cell Arrival Rate	*CAR*	167 cells/s.	65.105 c./s.
Cell inter-Arrival Time	*1/CAR*	6 ms.	15 µs.
Ancho de Banda de Enlaces	*AB*	155,52 Mbps.	622 Mbit/s.
Cell Slot Rate	*CSR*	353.208 cell/s.	1.412.648 cell/s.
Tiempo de servicio por célula	*1/CSR*	2,83 µs.	0,70 µs.
Periodo de tiempo de actividad	T_{on}	0,96 s.	1 s.
Nº medio de células en el estado T_{on}	X_{on}	160 cells	65.105 cells
Periodo de tiempo de inactividad	T_{off}	1,69 s.	2 s.
Nº medio de slots vacíos en el estado T_{off}	X_{off}	596.921 cell slots	2.825.296 cell slots

A continuación se presentan algunos de los resultados más destacables obtenidos en las simulaciones realizadas. Se destaca que todas estas simulaciones son realizadas de forma local sobre un ordenador que simula las topologías de VPN comentadas en el *Capítulo 7*, en el que puede elegirse el número de fuentes, el número de receptores, los conmutadores no activos y los AcTMs que constituyen la VPN. Además puede optarse por los valores de parámetros de tráfico presentados en la *Tabla 12.1* anterior.

Las evaluaciones presentadas se centran en los resultados más significativos y que argumentan las ventajas de TAP como son la recuperación de PDU que se perderían por congestión y además cuidando que todas esas retransmisiones no afecten al rendimiento de la red. Para ello se estudia primero el efecto del CAR sobre el índice de recuperaciones de PDU congestionadas. En segundo lugar se comprueba el efecto del T_{off} sobre el índice de recuperaciones. Por último, se presentan los resultados de las simulaciones aisladas del algoritmo QPWFQ sobre las colas de entrada de los conmutadores activos.

12.5.1. EVALUACIÓN DEL EFECTO DEL CAR SOBRE EL ÍNDICE DE RECUPARACIONES

En este caso, la configuración básica (punto-a-punto) de la VPN estudiada consiste en 3 conmutadores ATM activos con un solo emisor y un receptor.

La *Figura 12.7* muestra el efecto experimentado al variar el CAR entre 86 y 2.667 células por segundo (33.000 bps hasta 1 Mbps respectivamente). En esta simulación se ha fijado la probabilidad de congestión a 10^{-3}. Se usa un buffer de 3.000 octetos y la DMTE almacena 2 PDU de 1.500 bytes para cada conexión.

El valor de PCR es de 64 Kbps (167 cells/s.); T_{on} = 0,96 s.; y T_{off} = 1,69 s, y, sobre el total de las 50 PDU descartadas por congestión, 50 PDU son recuperadas por TAP. También, cuando el CAR=56 Kbps y 33 Kbps, TAP recupera todas las PDU congestionadas. Así, el rendimiento es optimizado (50 PDU recuperadas de 50 PDU congestionadas) porque todas las PDU perdidas son recuperadas y no se producen fallos de DMTE, ya que todas las PDU solicitadas se encuentran en la memoria DMTE del conmutador anterior cuando se solicita la retransmisión.

Como puede observarse en la *Figura 12.7*, cuando la velocidad de llegada de células es baja, el índice de PDU recuperadas es óptima. Cuando el CAR incrementa, 256 Kbps, TAP recupera 48 de las 50 PDU, pero 2 de las PDU perdidas no son solicitadas porque el protocolo detecta insuficiente tiempo de inactividad (T_{off}) para poder realizar la retransmisión y, bajo un escenario en el que se suponen un número de fuentes de *background* que superan la capacidad del enlace. Podemos ver cómo el número de NACK no enviados (PDU no solicitadas) es mayor cuando el valor de CAR incrementa. De este modo, la red no será sobrecargada con retransmisiones inútiles cuando no existe garantía de éxito en la recuperación al no disponer de suficiente tiempo de inactividad agregado.

Se destaca que a elevados valores de CAR (1 Mbps), el número de PDU recuperadas es de 47 y también en este caso, las 3 PDU no recuperadas no han sido solicitadas para evitar los potenciales fallos de DMTE. De este modo, podemos ver que el *goodput* es también optimizado a elevada velocidad, siempre y cuando el número de fuentes con GoS no exceda la capacidad de servicio CSR (ver *Capítulo 11*). Debe destacarse que se han realizado numerosas simulaciones para estudiar este comportamiento y todas han acabado aportando resultados similares, demostrando que TAP es capaz de recuperar un importante número de PDU que experimentan congestiones en los conmutadores y que acabarían perdiéndose en la red sin la existencia de TAP, ya que los mecanismos estándares de la tecnología ATM no se encargan de solventar los problemas relativos a las pérdidas en la propia red, sino que son detectadas en los receptores y recuperadas mediante retransmisión extremo-extremo con el consiguiente efecto sobre el rendimiento de la red, tal como se argumenta en el *Capítulo 8.*

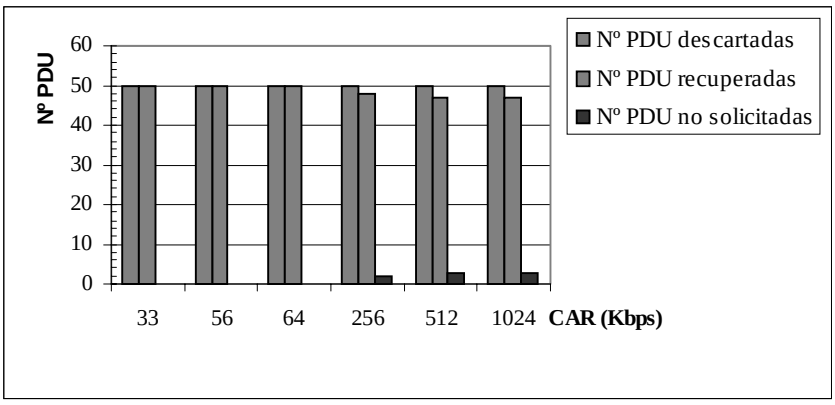

Figura 12.7*. Número de PDU recuperadas en función del CAR*

Se ha estudiado también el índice de recuperaciones sobre otros escenarios consistentes en *1* nodo fuente, *1* conmutador ATM activo, *n* conmutadores no activos y *1* nodo destino. Cuando llega un NACK a un conmutador no activo, éste también transfiere la célula RM al siguiente conmutador. Cuando la RM llega al conmutador activo éste usa la memoria DMTE para retransmitir la PDU solicitada. Este escenario es el mismo que el anterior, sólo que el número de conmutadores no activos varía. En esta configuración hemos simulado el protocolo con varios conmutadores no activos y los resultados obtenidos no cambian respecto a la configuración básica. Sólo varía el retardo en las transmisiones debido a los tiempos de propagación, pero el índice de PDU recuperadas se mantiene como en el caso básico.

Otros escenarios estudiados presentan una configuración punto-multipunto consistente en *1* nodo fuente, *1* conmutador ATM activo, *n* conmutadores no activos y *n* nodos destino. Los resultados obtenidos son similares al escenario básico punto-a-punto, sólo varía el número de nodos destino en las conexiones multipunto, obteniéndose resultados similares a los mostrados en la *Figura 12.7*.

Los resultados de la *Figura 12.7* varían al actuar sobre los diversos parámetros de la simulación como es de esperar. Sin embargo, uno de los aspectos que más afecta al índice de recuperación de pérdidas debidas a congestiones son los valores fijados de T_{on} y T_{off} de forma que, a medida que no se dispone de suficiente tiempo de inactividad agregado, el índice de PDU recuperadas desciende proporcionalmente. Este efecto se estudia en el siguiente apartado.

12.5.2. EFECTO DEL T_{off} SOBRE EL ÍNDICE DE RECUPERACIÓN DE PDU

La *Figura 12.8* muestra los resultados obtenidos al variar el tiempo de inactividad (T_{off}) entre 0,1 y 2 segundos. En este caso se usaron 10 fuentes ON/OFF sobre un enlace con ancho de banda de 25 Mbps. Cada fuente genera 500 PDU con un PCR=1 Mbps. Se fijó el Cell Loss Rate al 5% sobre el número total de las PDU emitidas y, en la figura, se ha mantenido constante el valor de 25 PDU congestionadas. Como puede observarse, cuando el T_{off} agregado es suficiente (0,5 s.), todas las PDU congestionadas son recuperadas (25 recuperaciones de 25 congestiones). Cuando el tiempo T_{off} es menor de 0,5 s. las PDU recuperadas bajan a 12 PDU cuando T_{off}=0,3 s., y a 3 PDU recuperadas cuando T_{off}=0,1 s. De este modo, cuando el tiempo T_{off} es insuficiente, el número de PDU irrecuperables crece, pero TAP garantiza el goodput ya que las PDU irrecuperables no son solicitadas para evitar sobrecargar la red (tal como se ha explicado en el apartado anterior). Por tanto, tal como se comprueba en la *Figura 12.8*, el hecho de no disponer de tiempo de inactividad no afecta negativamente al protocolo ni a la red, ya que en esta situación las PDU no serán solicitadas desde el AcTMs congestionado al conmutador previo. Como ya se ha comentado también en capítulos anteriores, ésta es una de las funciones del agente RCA.

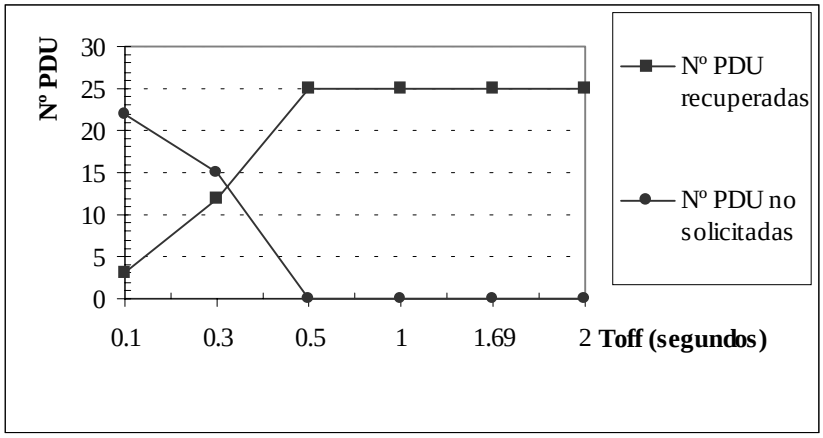

Figura 12. 8. *Efecto de la variación de Toff*

Se ha estudiado también el efecto de la variación del T_{off} sobre otros escenarios donde se alteraron diversos parámetros de simulación como el número de fuentes, el número de destinatarios, el ancho de banda de los enlaces y diversos PCR y nodos AcTMs y, en todas ellas, el efecto de la variación del T_{off} ha terminado dando resultados similares al mostrado en la *Figura 12.8*, demostrándose que la intuitiva idea de atender las retransmisiones aprovechando los tiempos de inactividad de las fuentes y de los enlaces permite aprovechar el goodput en la red.

12.5.3. OTROS ASPECTOS DESTACABLES

Como ya se demostró en el *Capítulo 11*, al incrementar el número de fuentes de background (sin requerimientos de GoS) en las simulaciones se acaba también afectando al índice de recuperación de PDU. Esto es así por el hecho que al aumentar el tráfico en los conmutadores con fuentes de background en realidad lo que se produce es un consumo elevado de la capacidad del enlace y, por tanto, del T_{off} agregado, lo que acaba dando resultados similares a los presentados en la *Figura 12.8*.

Como era de esperar, las simulaciones demuestran también que las variaciones en el tamaño del buffer

afectan al índice de recuperaciones de PDU. Cuanto menor es el tamaño del buffer, mayor es la probabilidad de congestiones en el mismo y, por tanto, incrementa el número de retransmisiones realizadas, aunque el índice de recuperaciones con éxito depende sobre todo de los parámetros estudiados en las *Figuras 12.7* y *12.8*. En el caso de tamaño de PDU de 1.500 octetos, se he comprobado que un tamaño de buffer de 20 Kbytes (427 células) acaba dando resultados aceptables.

El tamaño de PDU usado también afecta a los resultados obtenidos, ya que al emplear PDU de mayor tamaño incrementa también la probabilidad de congestiones en el buffer. Es decir, si se emplea un buffer de 20 Kbytes y PDU de tamaño cercano (o superior a 20 Kbytes) acaba provocando descartes de estas PDU que, como sabemos, pueden tener un tamaño máximo de hasta 65.536 octetos. Por este motivo es importante buscar el punto de equilibrio entre los tamaños de PDU y el tamaño reservado para el buffer de los AcTMs.

Otro aspecto de interés es el efecto del umbral fijado sobre el buffer por el algoritmo EPDR. Aunque el valor del umbral depende directamente también del tamaño del buffer y de las PDU, en la mayor parte de situaciones se obtiene un buen comportamiento cuando el buffer se sitúa en entre el 95% y el 97% del tamaño total del buffer. Actuar sobre este valor permite amortiguar el efecto de la fragmentación de las PDU, pero también aumenta el número de retransmisiones realizadas. Recordamos que la función de ajuste del valor del umbral es responsabilidad del agente programable CCA.

Como se ha destacado en el *Capítulo 10*, otro aspecto que merece comentario es el tamaño de la DMTE empleada. En este caso, como en el del buffer, es necesario buscar un adecuado punto de equilibrio entre el tamaño de la memoria dinámica y el de las PDU. Intuitivamente se ve que el tamaño de la DMTE debe ser mayor cuanto mayor es el tamaño de las PDU. Además, el tamaño de la DMTE también depende del parámetro GoS que se defina para cada fuente; es decir cuando se define una fuente con GoS con valor a 2, esto quiere decir que se almacenan 2 PDU en la DMTE para esa fuente. A medida que se incrementa el parámetro GoS, mayor es la garantía de servicio aportada a las fuentes; sin embargo, mayor es la necesidad de memora DMTE. La misma discusión puede hacerse con respecto al número de fuentes privilegiadas que se definen; es decir, cuantas más fuentes con GoS se empleen, mayor será el requerimiento de memoria. Por tanto, para que el índice de recuperaciones de la DMTE sea aceptable es necesario encontrar una situación de equilibrio entre el tamaño total de DMTE, el tamaño de cada una de las PDU, el parámetro de GoS de cada fuente, y el número de fuentes privilegiadas. Ajustando adecuadamente los valores de todos estos parámetros se logran también aceptables índices de recuperación de PDU.

En resumen, con las simulaciones realizadas se ha comprobado que la arquitectura TAP distribuida y activa aprovecha las ventajas de los conmutadores AcTMs. Así, se ha verificado que es posible recuperar un importante número de PDU con un aceptable tamaño de memoria DMTE y una razonable complejidad añadida en los conmutadores activos soportada por agentes software. Estas simulaciones demuestran también que la idea intuitiva de aprovechar los tiempos de silencio en las fuentes ON/OFF es cierta consiguiendo mejorar el comportamiento y QoS en las redes ATM que soportan fuentes que requieren GoS.

12.5.4. EFECTOS DEL ALGORITMO QPWFQ SOBRE COLAS DE ENTRADA Y BUFFER

En este apartado se presentan los detalles de funcionamiento del algoritmo QPWFQ, implementado como parte del agente WFQ del SMA-TAP. Este algoritmo es explicado en los *Capítulos 4* y *10* y, a continuación, se presenta su entorno de ejecución en el simulador y se comentan algunos de los resultados más significativos que se han obtenido. Se destaca que en este caso vamos a estudiar el comportamiento del algoritmo de forma aislada; es decir, no interviene ningún otro elemento del SMA. El escenario de simulación, por tanto, está formado por tres conmutadores activos y a cada uno de ellos llegan cuatro VPI/VCI, de forma que las cuatro entradas del conmutador A se multiplexan en una de las cuatro entradas del conmutador B. A su vez las cuatro entradas del conmutador B se multiplexan a su salida que es una de las entradas del conmutador C, que dispone también de cuatro entradas y una sola salida. De este modo, se pueden elegir varias fuentes privilegiadas y a su vez simular el efecto de la multiplexación y el de las fuentes de *background*.

Cada uno de los tres AcTMs dispone de sus cuatro colas de entrada, de su correspondiente buffer y de su propia DMTE. Nuestra intención es estudiar las congestiones, el número de retransmisiones y su efecto sobre el rendimiento del cada conmutador. El escenario de simulación puede observarse en la *Figura 12.9*, donde se muestra el aspecto del conmutador A, con sus cuatro fuentes y colas de entrada, con una asignación de pesos en las colas, el buffer y la salida multiplexada hacia el VPI 5.

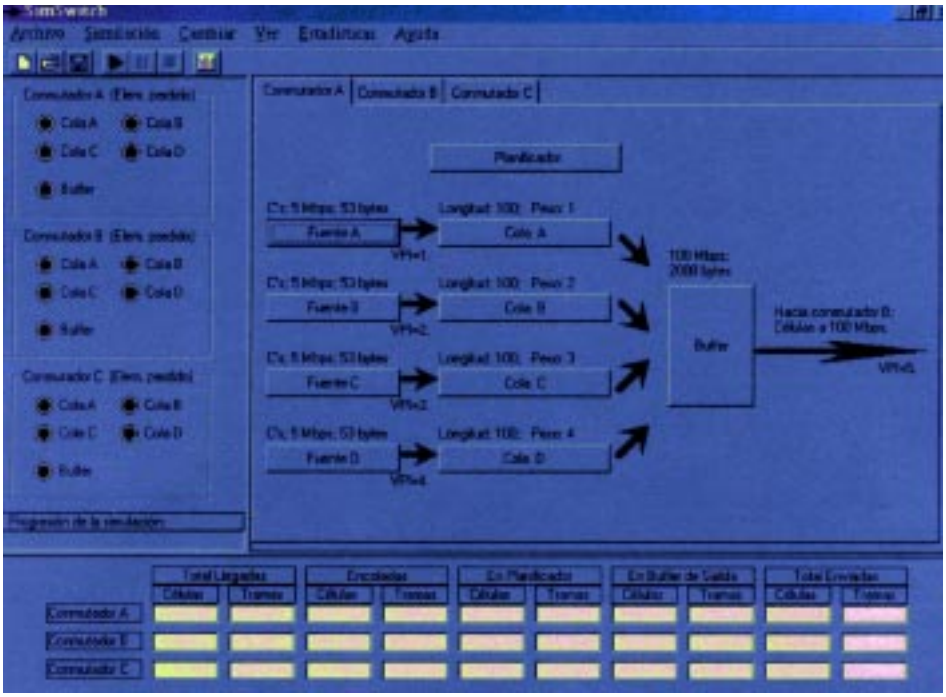

Figura 12.9. *Entorno de simulación del algoritmo QPWFQ sobre un AcTMs*

A continuación se comentan brevemente cada uno de los apartados del entorno de la *Figura 12.9*.

- La *barra de menús*, situada en la parte superior de la *Figura 12.9*, aporta seis opciones que permiten actuar sobre el simulador (*Archivo*, *Simulación*, *Cambiar*, *Ver*, *Estadísticas* y *Ayuda*).
- La *barra de botones*, situada bajo la barra de menús, se compone de seis botones cada uno programado con diversas acciones sobre las simulaciones (*Nuevo*, *Abrir configuración*, *Guardar configuración*, *Iniciar simulación*, *Congelar simulación*, *Detener simulación* y *Estadísticas de simulación*).
- *Zona de descripción* de los conmutadores del escenario, que se muestra en la parte central de la *Figura 12.9*. Pueden observarse tres fichas, donde cada una corresponde a cada uno de los conmutadores con sus diferentes componentes que son los siguientes:
  - ✓ *Fuentes* dispuestas para la generación del tráfico que llega a cada conmutador. Cada fuente tiene los siguientes parámetros: *Tipo* de fuente con GoS o no (generan PDU o células); *CAR* del tráfico que puede ser desde 1 a 100 Mbps; *Longitud* de las PDU (desde 61 a 65.535 bytes); y *Estado* de la fuente (activa o inactiva).
  - ✓ *Colas* que representan la llegada de las transferencias desde las fuentes. Tiene como parámetros modificables el *peso* y la *longitud* de la cola.
  - ✓ *Buffer* que representa al buffer de los AcTMs sobre los que confluye el trafico de las colas. Tiene como parámetros modificables el *tamaño* (comprendido entre 100 y 100.000 bytes); la *velocidad* de conmutación (de 1 a 100 Mbps); y la velocidad de salida de las unidades procesadas o capacidad de servicio (de 1 a 100 Mbps).
  - ✓ El *planificador* representa a las cola de turnos del algoritmo.
- *Zona de información de estado del flujo*, situada en la parte inferior de la *Figura 12.9*, presenta, en tiempo de simulación, la información de las unidades de transferencia (tramas o células) procesadas en cada uno de los conmutadores.
- *Zona de progresión de la simulación*, situada en la parte inferior izquierda de la *Figura 12.9*, donde puede seguirse la evolución de las PDU y células a lo largo de las colas o buffers en cada conmutador, informando visualmente cuando se producen pérdidas de PDU por congestión del buffer.

En cuanto a las estadísticas se han implementado tres posibilidades:

- *Peticiones de retransmisión* para obtener una representación gráfica del número de solicitudes de

retransmisión que ha realizado cada uno de los AcTMs hacia el conmutador previo, en función del tiempo de simulación.

- *Retransmisiones atendidas* para obtener una representación gráfica del número de PDU retransmitidas por cada conmutador hacia el conmutador siguiente del escenario y en función del tiempo de simulación.
- *Índice de Productividad* de cada conmutador, que muestra el número de células por crédito (intervalo de tiempo en el que se procesa un tráfico proporcional al tráfico de entrada) que cada conmutador envía a su puerto de salida sobre el que multiplexa todas las entradas y expresado en función del tiempo. Esta estadística muestra, por tanto, el goodput de cada conmutador teniendo en cuenta, tanto las transmisiones, como las retransmisiones procesadas.

En este caso se simulan las congestiones de forma realista, es decir, sólo se realiza una retransmisión de PDU cuando se pierde, y una PDU se puede perder en los dos casos siguientes:

- Se intenta introducir en una cola una PDU cuando la cola está llena en ese momento. Las colas están llenas cuando la velocidad de conmutación en el buffer es menor que la velocidad total de llegada de tráfico a éste.
- Se intenta introducir en el buffer una PDU que no cabe.

Seguidamente se muestran los resultados de algunas de las simulaciones realizadas.

La *Figura 12.10* presenta la productividad de los conmutadores en un escenario en el que no se ha producido ningún tipo de congestión. En este caso los parámetros de la simulación han sido los siguientes: Todas las fuentes generan en CAR de 5 Mbps, los pesos de cada cola se corresponden con el CAR de su correspondiente fuente. En cada cola tienen cabida 100 unidades de transferencia (PDU o células), y los tres buffers, con una capacidad de 2.000 octetos, conmutan a 100 Mbps. Se realizó una simulación de 8,56 ms., de forma que el conmutador A fue capaz de procesar 400 células, el conmutador B 415 células y el C 448 células. Con estos parámetros de tráfico no se acaba produciendo ninguna congestión en los buffers y, como puede observarse en la *Figura 12.10*, el volumen de tráfico es constante a lo largo de todo el tiempo de simulación, salvo al final. Este efecto se produce por el corte de tráfico desde el conmutador A que provoca al final de la simulación una bajada en el tráfico servido a los conmutadores B y C. En este caso las estadísticas de retransmisiones mostrarían gráficas completamente planas al no haber congestiones.

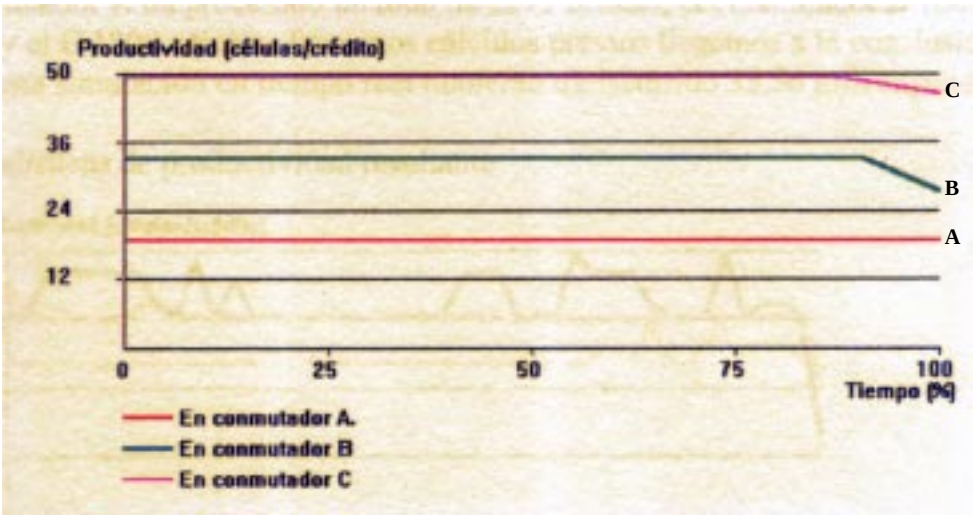

Figura 12.10. *Productividad en un escenario sin congestiones*

La *Figura 12.11* muestra el resultado de una simulación con congestiones en el buffer de los conmutadores. Los parámetros del escenario de esta simulación se explican a continuación.

- Conmutador A: se han introducido tres fuentes de *background* de 5 Mbps cada una, y una fuente con GoS que genera PDU de 530 bytes a 10 Mbps. Las colas tienen una capacidad de almacenamiento de 100 unidades, y pesos que se corresponden con la velocidad de las fuentes. El buffer tiene una capacidad de 20 Kbytes, velocidad de conmutación de 100 Mbps y la velocidad de salida es de 30 Mbps. Con estos datos no se prevén congestiones en el conmutador.

- Conmutador B: con tres fuentes de *brackground* idénticas al conmutador A, lo mismo que las colas. Sin embargo, el buffer es de 15 Kbytes con velocidad de conmutación de 100 Mbps y velocidad de salida o capacidad de servicio de 35 Mbps. En este caso el buffer se llenará progresivamente.
- Conmutador C: las fuentes de *background* y las colas son idénticas a los conmutadores A y B. Pero en este caso el buffer es de 2 Kbytes con una velocidad de conmutación de 100 Mbps y capacidad de servicio de 35 Mbps. Debido a la escasa capacidad del buffer se prevé el llenado progresivo del buffer y una elevada probabilidad de congestiones debido al escaso tamaño del buffer.

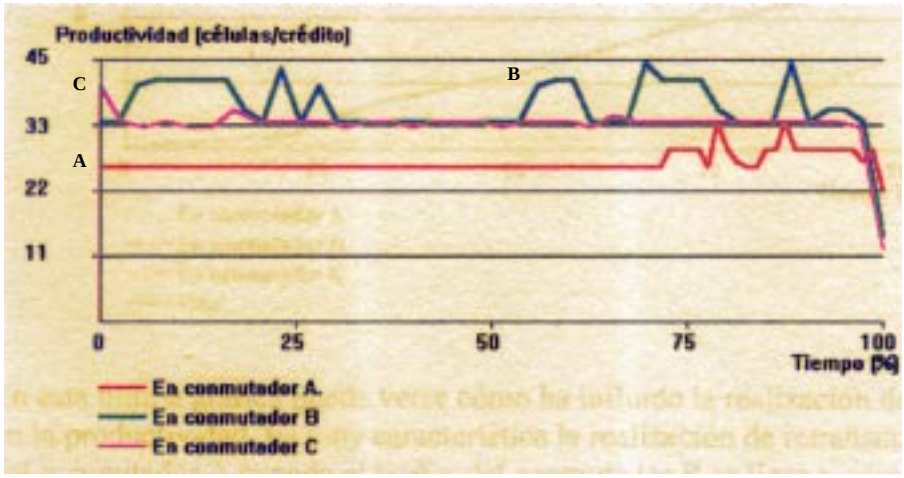

Figura 12.11. *Índice de productividad con buffer congestionado*

En este escenario de simulación el conmutador A procesa un total de 2.273 células, el B 1.645 células y el C 1.300 células, todo ello durante 38,56 ms. Puede observarse un tráfico casi constante en el conmutador C, mientras en el conmutador B aparecen importantes fluctuaciones de productividad, que serán justificadas en gráficas siguientes y que son debidas a las retransmisiones solicitadas desde el conmutador C. En el caso del conmutador A al principio no se produce variación; sin embargo, al final de la simulación el flujo constante experimenta picos debidos a las solicitudes de retransmisión recibidas desde el conmutador B.

La gráfica de la *Figura 12.12* presenta la estadística de petición de retransmisiones realizadas por cada uno de los conmutadores activos. Como puede observarse, en el caso del conmutador C las retransmisiones se producen casi desde el inicio de la simulación, debido al pequeño tamaño de buffer que provoca la pérdida de las PDU. Sin embargo, el conmutador B no realiza retransmisiones hasta el 75% del tiempo de simulación que es cuando comienza a experimentar las congestiones debidas a la sobrecarga de tráfico provocada por las solicitudes de retransmisión recibidas desde el conmutador C. Podemos comprobar que en la simulación se han perdido cinco PDU (se solicitan 22 retransmisiones y se retransmiten 17 PDU). Puede verse que las retransmisiones que se pierden pertenecen al conmutador B que sólo sirve 10 PDU de las 15 que le solicita el conmutador C. El conmutador A no realiza ninguna petición de retransmisión, y si fuese así, esto sería indicativo de pérdidas, ya que el conmutador A no tiene por encima ningún otro conmutador.

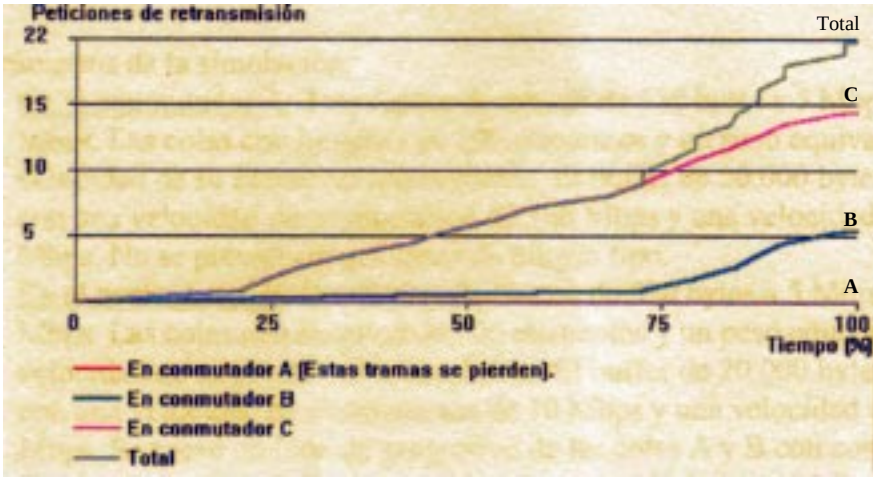

Figura 12.12. *Estadística de solicitudes de retransmisión*

A continuación se estudian las estadísticas relativas a las retransmisiones atendidas por cada uno de los conmutadores. La *Figura 12.13* nos ayuda a entender cómo influyen la atención de las retransmisiones en la productividad representada en la *Figura 12.11*. Es muy significativo el hecho de que en la *Figura 12.13* el conmutador A comienza a atender solicitudes de retransmisión que provienen del conmutador B cuando su buffer se congestiona (debido a las solicitudes de retransmisión provenientes del conmutador C). A medida que se empiezan a atender las retransmisiones en el conmutador A en la *Figura 12.13*, se comienzan a producir las fluctuaciones de productividad en el conmutador A que se observaron en la *Figura 12.11*. En la *Figura 12.13* puede observarse cómo el conmutador C no atiende ninguna solicitud de retransmisión por ser el último de la red. Hemos de destacar que este efecto de sobrecarga sobre los conmutadores que realizan las retransmisiones se produce porque estas simulaciones del algoritmo QPWFQ se han realizado para estudiar su comportamiento aislado del resto de mecanismos de TAP, y por esto no actúa el control de tiempos de inactividad antes de responder a las retransmisiones.

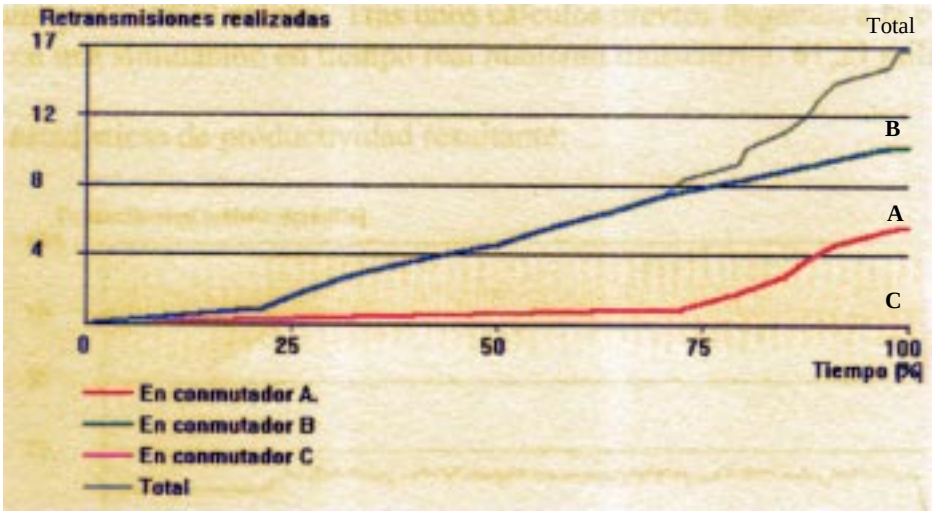

Figura 12.13. *Estadísticas de retransmisiones servidas*

A continuación se presenta un último escenario de simulación donde se estudian las congestiones sobre las colas de entrada. En este caso los parámetros de la simulación son los siguientes:

- Conmutador A: Se emplean tres fuentes con GoS de 530 bytes y un CAR de 5 Mbps. La cuarta fuente es de background a 50 Mbps. Las colas de entrada son de 100 elementos y un peso equivalente a la velocidad de sus correspondientes fuentes. El buffer es de 20 Kbytes de capacidad y una velocidad de conmutación de 100 Mbps y un capacidad de servicio de 100 Mbps. En este caso no se prevén congestiones.
- Conmutador B: Se introducen dos fuentes privilegiadas de 530 octetos y CAR de 5 Mbps. Una tercera fuente es de background con un CAR de 50 Mbps. Las colas son idénticas a las del conmutador A. El buffer es de 20 Kbytes de capacidad, con una velocidad de conmutación de 10 Mbps y una capacidad de servicio de 100 Mbps. En este caso se intuye un llenado progresivo de las colas A y B con congestión que corresponden a las dos fuentes privilegiadas.
- Conmutador C: Se usan dos fuentes privilegiadas de 530 bytes a 5 Mbps, y una tercera de relleno a 10 Mbps. Las colas son iguales a los conmutadores A y B. El buffer es de 2 Kbytes de capacidad y una velocidad de 10 Mbps, con una capacidad de servicio de 100 Mbps. Se prevé un llenado progresivo de las colas A y B y congestiones probables en estas colas debido a la baja velocidad de conmutación del buffer.

Se ha simulado este escenario durante 61,23 milisegundos, de forma que el conmutador A procesó 9.387 células, el conmutador B 1.747 células y el conmutador C 5.570 células.

La *Figura 12.14* presenta la gráfica de productividad de los conmutadores, observándose cómo en el conmutador A queda caracterizado el tráfico a ráfagas propio de la transmisión de las PDU al existir tres fuentes privilegiadas. También se observa que los conmutadores B y C transmiten muy poca cantidad de tráfico debido a la baja velocidad de conmutación que tienen. Puede observarse cómo el volumen de tráfico general se incrementa al llegar al 25 % del tiempo de simulación, debido al inicio de congestiones y a la aparición de retransmisiones como podrá comprobarse en las dos siguientes figuras.

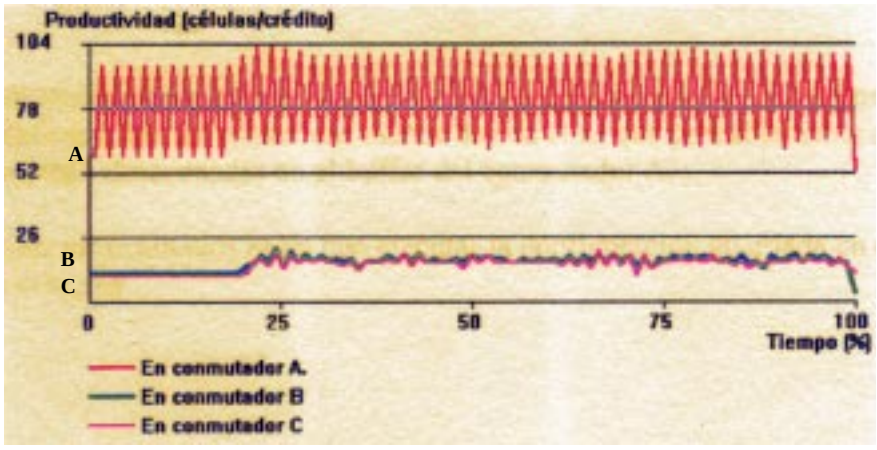

Figura 12.14. *Estadística de productividad en congestiones sobre las colas*

La *Figura 12.15* presenta la estadística relativa a las solicitudes de retransmisión realizadas por los tres conmutadores. Puede observarse cómo sólo el conmutador B realiza solicitudes de retransmisión a partir del 25 % del tiempo de simulación debido a la congestión experimentada en sus colas de entrada y motivada por la baja velocidad de conmutación del buffer y a la elevada fuente de *background* de 50 Mbps introducida.

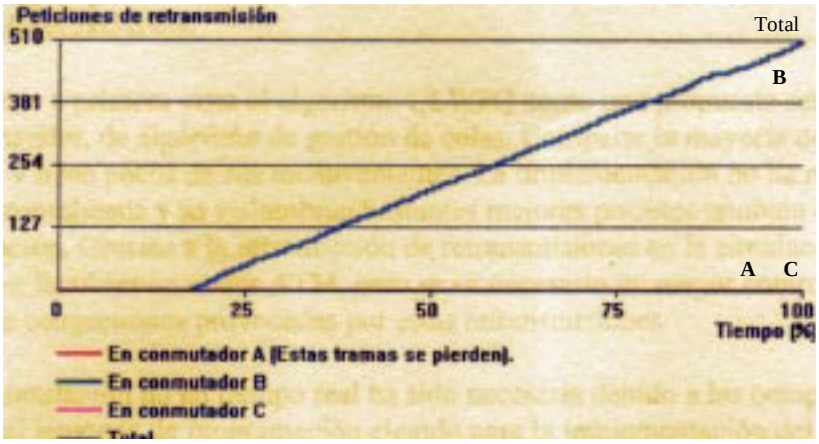

Figura 12.15. *Peticiones de retransmisión realizadas por los conmutadores*

Por último, se estudia en este tercer escenario la estadística de retransmisiones realizadas por los conmutadores. En la *Figura 12.16* puede comprobarse cómo las retransmisiones realizadas siguen el mismo ritmo que las peticiones de retransmisión representadas en la *Figura 12.15*. En la *Figura 12.15* es el conmutador B el que solicita las retransmisiones, mientras en la *Figura 12.16* se observa cómo es el conmutador A el que se encarga de realizar las retransmisiones. Comparando las dos figuras podemos comprobar que se han realizado 510 peticiones de retransmisión desde B, mientras han sido retransmitidas 508 de ellas desde A, debido al corte de la simulación desde el conmutador A.

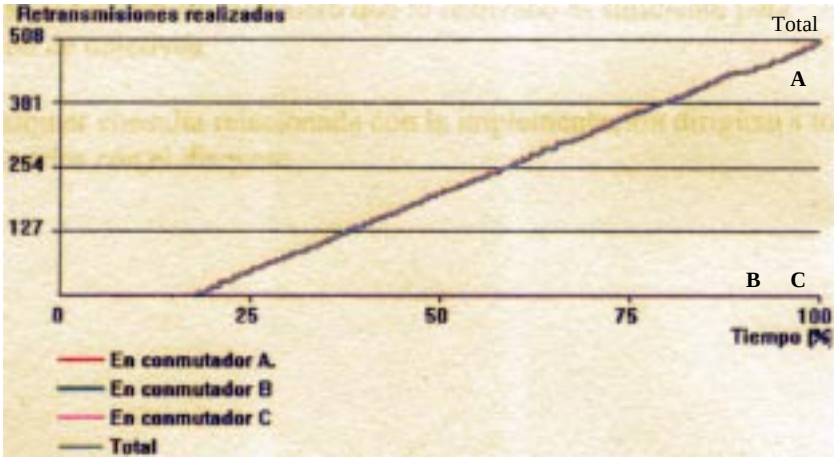

Figura 12.16. *Número de retransmisiones realizadas*

12.6. CONCLUSIONES

El lenguaje Java y su JVM ofrecen atractivas ventajas para el desarrollo de protocolos de comunicaciones portables y extensibles. Además, las propias características del lenguaje permiten también la incorporación en esos protocolos de las premisas necesarias para el desarrollo de sistemas multiagente, ya que la propia extensibilidad aportada por Java permite desarrollar agentes programables. El propio lenguaje Java ha sido identificado como una de las herramientas más adecuadas para implementar agentes, sus interacciones y comunicación con otros agentes en los SMA. El relativo corto periodo de vida del lenguaje no permite tener datos acertados sobre su propia evolución en cuanto al rendimiento aportado por Java con respecto a otros lenguajes en el desarrollo de protocolos; sin embargo, la mejora de sus nuevas versiones permite adelantar que esto beneficiará al rendimiento a medida que evolucionen el propio compilador de Java y la JVM.

Aunque la implementación del protocolo SMA-TAP no es el principal objetivo de esta tesis, hemos presentado la especificación, las decisiones de diseño y algunos de los detalles de implementación del prototipo de simulador de TAP, desarrollado en lenguaje Java con JDK 1.1 y que nos ha servido para evaluar algunos de los aspectos de rendimiento y comportamiento de TAP, a la vez que ha sido la excusa para investigar y experimentar sobre la ingeniería de protocolos de comunicaciones basados en agentes software. Los experimentos realizados han permitido también identificar interesantes vías de investigación futuras para el enriquecimiento e implementación de un simulador más avanzado en el que el SMA sea concluido e incorpore las posibilidades de CORBA y RMI con la intención de poder simular el comportamiento de TAP en una red real sobre la que realizar el estudio de comportamiento del protocolo con tráfico real.

El simulador de TAP se ejecuta sobre un ordenador aislado donde se pueden elegir determinados escenarios con el objetivo de conocer el comportamiento de TAP en un entorno local. De este modo se ha estudiado el comportamiento del algoritmo QPWFQ sobre las colas de entrada en los conmutadores, y se ha comprobado que TAP recupera un importante número de PDU que de otro modo se perderían en situaciones de congestión. También se ha comprobado que se satisface la idea intuitiva de aprovechar los periodos de inactividad de las fuentes y de los enlaces para atender las retransmisiones en los conmutadores activos. Se ha analizado además el comportamiento del algoritmo QPWFQ de forma aislada, comprobándose que sólo con este algoritmo y la DMTE se puede conseguir recuperar un importante número de PDU congestionadas.

REFERENCIAS

[1] Deepika Chauhan, JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation," *ECECS Deparment Thesis, U. Cincinnati,* (1997) *http://www.ececs.uc.edu/~abaker/JAFMAS/*

[2] Sun Microsystems, "Java Whitepaper," *http://java.sun.com/doc/overview/java/index.htm*, (1996).

[3] L. Cardelli, "Obliq: A Language with Distributed Scope", *Technical Report, Digital Equipment Corporation, Systems Research Center*, May. 1995.

[4] General Magic, "Telescript Language Reference", *General Magic*, Oct. 1995.

[5] B. Krupczak, K. L. Calvert, and M. H. Ammar, "Implementing communications protocols in Java," *IEEE Communications Magazine*, pp. 93-99, (1998).

[6] B. Krupczak, K. L. Calvert, and M. Ammar, "Implementing protocols in Java: The price of portability," *Procs. IEEE INFOCOM*, (1998).

[7] J. Gosling, B. Joy, and G. Steele, "The Java languaje specification, v. 1.0," *Sun Microsystems*, (1996).

[8] Sun Microsystems, "The Java Virtual Machine specification, v. 1.0," *Technical Report* (1995).

[9] Jamie Jaworski, "Java 1.2 Al descubierto," *Ed. Prentice Hall* (1999).

[10] H. Jeon, C. Petrie and M. R. Cutkosky, "JATLite: A Java Agent Infraestructure with Message Routing," *IEEE Internet Computing,* (2000). *http://www-cdr.stanford.edu/ProcessLink/papers/JATL.html*

[11] José Luis González-Sánchez and Jordi Domingo-Pascual, "RAP: Protocol for Reliable Transfers in ATM Networks with Active Switches," *Technical Report UPC-DAC-1999-54. http://www.ac.upc.es/recerca/reports/INDEX1999DAC.html* (1.999).

[12] José Luis González-Sánchez and Jordi Domingo-Pascual, "RAP: Protocol for Reliable Transfers in ATM Networks with Active Switches," *International Conference on Communications in Computing (CIC'2000).*

[13] José Luis González-Sánchez and Jordi Domingo-Pascual, "TAP: Architecture for Trusted Transfers in ATM Networks with Active Switches," *ATM'2000 IEEE Conference on High Performance Switching and Routing Joint IEEE ATM Workshop 2000 and 3rd International Conference on ATM (ICATM'2000),* (2000).

[14] José Luis González-Sánchez and Jordi Domingo-Pascual, "Trusted and Active Protocol over a Distributed Architecture in ATM Networks with agents," *IEEE International Conference on Computer Communication and Networks (IEEE IC3N'2000),* (2000).

CAPÍTULO 13

LÍNEAS DE INVESTIGACIÓN FUTURA Y CONCLUSIONES FINALES

13.1. LÍNEAS DE INVESTIGACIÓN FUTURA

El proceso de desarrollo de esta tesis doctoral ha dado lugar a la identificación de nuevas líneas de investigación que lleven a la mejora de las posibilidades de la arquitectura propuesta. De este modo, este último capítulo tiene por objetivo destacar algunas de las líneas en las que prevemos dar continuidad a nuestras investigaciones.

ATM es considerada actualmente como una tecnología suficientemente implantada y madura; sin embargo, hemos tenido la ocasión de demostrar que aún quedan limitaciones por resolver y en esta línea se ha situado la arquitectura y el protocolo TAP con posibilidades, tanto para el tráfico ATM nativo, como para el generado por aplicaciones TCP/IP. No obstante, la propia tecnología ATM sigue evolucionando y dando lugar a nuevas clases de servicio y a interesantes propuestas de integración con los protocolos TCP/IP. La ingeniería de protocolos, por tanto, puede ofrecer aún nuevos mecanismos que los estándares no han cubierto, como la posibilidad de aportar el parámetro de GoS a las fuentes que tengan requerimientos de garantía de servicio. La propuesta de GoS ofrecida por TAP puede ser aprovechada por las fuentes ABR, UBR y también por la nueva CoS GFR, de forma que ésta es una de esas nuevas oportunidades de investigación.

Las posibilidades de las redes activas han sido también identificadas como otro atractivo campo de aplicación para las nuevas arquitecturas de comunicaciones. Aunque TAP es una propuesta novedosa en este campo, quedan aún por aplicar muchas ideas de las redes programables en el ámbito ATM. Así en TAP pueden incluirse nuevas características del paradigma de los agentes software como la movilidad de código, la inclusión de agentes inteligentes o la normalización acorde a las propuestas más extendidas como CORBA o FIPA.

Hemos podido comprobar cómo la arquitectura TAP no sólo está equipada con técnicas software, sino que, además, los AcTMs también requieren de una equipación hardware apropiada para poder desempeñar sus funciones. De esta forma, esta arquitectura puede ser también objeto de estudio desde el campo de las arquitecturas especializadas. Seguidamente se amplían todas estas futuras líneas de acción donde prevemos centrar nuestras próximas investigaciones.

13.1.1. APLICACIÓN A LAS CLASES DE SERVICIO ABR Y GFR

Como se ha explicado a lo largo de esta memoria de tesis, el parámetro de GoS propuesto está especialmente indicado para fuentes de tráfico tipo UBR y ABR que no tienen especiales requerimientos en cuanto a la velocidad de transmisión. Pero la GoS es también adecuada para las conexiones que, necesitando un cierto grado de fiabilidad, desean obtener un adecuado rendimiento de la red. De este modo, TAP puede ser un protocolo indicado, tanto para aplicaciones de datos, como para aplicaciones híbridas (datos y tiempo real simultáneamente) que requieren garantía en la información transmitida. Desde este punto de vista UBR

(por ejemplo, tráfico *best effort* de la tecnología TCP/IP) es el tipo de fuente más indicado, pero también el tráfico ABR para el que deben garantizarse ciertos parámetros como PCR, MCR y ECR (ver *Capítulo 1*).

Como sabemos, se han empleado las fuentes ON/OFF (que modelan adecuadamente el tráfico a ráfagas, como UBR) para estudiar el comportamiento de TAP. Sin embargo, otro de nuestros objetivos futuros es poder modelar también adecuadamente el tráfico ABR incluyendo los parámetros de tráfico estáticos (PCR, MCR y CLP), los dinámicos (ECR, CI, NI, longitud de colas y ACR) e, incluso, los parámetros CTD y CDV (que no están especificados para ABR, pero se espera de la red que no experimente excesivo retardo). De este modo, TAP podría responder adecuadamente con estos parámetros, al que se añadiría el requerimiento de GoS. Es por tanto un objetivo a corto plazo el de poder estudiar el rendimiento del protocolo con esta caracterización, y poder realizar su evaluación con tráfico real, además de las simulaciones realizadas con las fuentes ON/OFF.

Otra de las clases de servicio ATM identificadas en el *Capítulo 1* es GFR, propuesta para las aplicaciones que generan tramas en lugar de células nativas ATM. Desde este punto de vista TAP puede responder perfectamente a esta nueva CoS, ya que la gestión de PDU realizada por el protocolo se ajusta perfectamente a los objetivos definidos en GFR. Así, se dará respuesta a aplicaciones sin requerimientos de tiempo real y con parámetros como PCR, MCR, MBS y MFS que necesitan tener garantizada una velocidad mínima de tramas. La propuesta GFR permite el descarte de tramas y no se especifica QoS, aunque la variante GFR2 puede ser usada para, mediante el bit CLP, rechazar tramas en las situaciones de congestión. En este caso, TAP podría encargarse de dar solución a las situaciones de congestión en las que GFR no ha especificado ninguna acción específica que no sea la retransmisión extremo-extremo. Nuestra investigación futura se centrará también en las posibilidades de evaluar el comportamiento de TAP en las aplicaciones que tengan estos requerimientos.

13.1.2. SOPORTE DE MOVILIDAD DE CÓDIGO EN EL SMA

Aunque se han identificado las posibilidades de la movilidad de código en el caso de TAP, el prototipo desarrollado no implementa esta potencialidad, por lo que consideramos que se han de estudiar más profundamente esta interesante característica. Así, es necesario evaluar las posibilidades que ofrecen la recuperación de PDU perdidas por congestión a lo largo y ancho de todo el árbol de distribución que constituye la red en el caso de las conexiones punto-multipunto y multipunto-multipunto.

Aunque parece claro que en nuestro caso no tiene excesivo interés la movilidad de los agentes en la red, si que podrán obtenerse posibilidades para TAP en el caso de poder mover cierto código que permita optimizar la localización de las PDU en los nodos activos antes de llegar a las fuentes de tráfico, evitando así la implosión sobre las fuentes. Es necesario evaluar detenidamente esta posibilidad y analizar su viabilidad de implementación. Del mismo modo es de interés considerar también los aspectos relacionados con la distribución multicast y la posible comunicación entre agentes del mismo nivel de forma similar a las propuestas existente para IP en *Reliable Multicast Protocol* y similares.

Por otro lado, puede considerarse la inclusión de características relativas a la inteligencia en alguno de los agentes (como por ejemplo RCA) para responder a las retransmisiones en función de aspectos concretos, como puede ser la de evitar la implosión sobre las fuentes en función del estado de la propia red. Del mismo modo podría analizarse también la posibilidad de priorizar las retransmisiones de PDU pertenecientes a las conexiones privilegiadas con respecto a las fuentes que no lo son.

13.1.3. DESARROLLO DE PROTOTIPO DE RED TAPS NORMALIZADO

Como se ha indicado en el *Capítulo 12*, el prototipo desarrollado para las simulaciones de TAP no se ha especificado para funcionar en un entorno de red, por lo que consideramos que puede ser interesante analizar la posibilidad de simular el protocolo en una red real y, para ello, pensamos en implementar TAPS incluyendo RMI o CORBA para aprovechar las ventajas de los objetos distribuidos.

Con los objetos distribuidos puede alcanzarse el máximo nivel de abstracción, donde las llamadas a métodos de objetos remotos (residentes en otros elementos de la red, o en otros procesos de un mismo ordenador aislado) se ejecuten de la misma forma que las llamadas a métodos de objetos locales que residen en el mismo espacio de direcciones del objeto llamante. RMI (*Remote Method Invocation*) permite a objetos Java llamar a métodos de otros objetos que están ejecutándose en otros ordenadores, como si fueran llamadas a objetos definidos localmente por la aplicación. En el caso de JDK 1.1, que es el entorno empleado para nuestros desarrollos, se podría realizar a través de las clases *java.rmi.** y *java.rmi.server.**.

El proceso por el que un objeto puede invocar métodos de un objeto remoto se divide en dos partes: la obtención de una referencia al objeto y la invocación propiamente dicha. Se podrán obtener referencias a

objetos a los que previamente se les haya declarado como remotos, es decir, que ofrecen la posibilidad de ser llamados remotamente. Esta característica es muy atractiva para la extensión de nuestras simulaciones a la ejecución del protocolo TAP sobre una red real, de forma que todos aquellos nodos que implementen TAP podrán soportar los agentes desarrollados, que trabajarán cooperativamente y de forma distribuida para realizar las funciones para las que han sido creados.

RMI, por tanto, nos ofrece la posibilidad de ejecutar en un entorno de red el protocolo TAP, pudiendo distribuirlo en nodos remotos, de forma que puedan definirse los nodos emisores y receptores y también los nodos AcTMs activos que se comportarán según lo especificado en capítulos anteriores.

Se trata por tanto de definir una interfaz remota (*RemoteCollection*) que declare los métodos que pueden ser invocados remotamente en los objetos que lo implementen. Cuando se desarrollan objetos que implementan esta interfaz (servidor) hay que asignarle un nombre y anotarlo en el registro RMI para que los clientes lo puedan localizar. El registro RMI es una asociación entre nombres de objetos servidores de interfaces y referencias a esos objetos. De este modo, cualquier cliente puede conectarse a un registro RMI (servicio basado en *sockets* TCP y puertos *well-known*, e implementado como una aplicación que puede estar ejecutándose en diversas máquinas, con un registro distribuido) y obtener las referencias a los objetos deseados a través de su nombre.

RMI aporta ventajas con respecto a los métodos más tradicionales para el procesamiento distribuido, de modo que se pueden aprovechar las grandes posibilidades de la POO como el polimorfismo y la ligadura dinámica. La invocación de métodos en objetos remotos es transparente a los programadores e independiente de los objetos. Además, las interfaces empleadas en RMI facilitan la modularidad, extensibilidad y reusabilidad de los desarrollos.

Aunque RMI aporta importantes características como las citadas, este entorno está claramente pensado para el desarrollo de pequeñas aplicaciones, por lo que podría pensarse en extender las especificaciones propuestas hasta CORBA (*Common Object Request Broker Architecture*). CORBA es un entorno estándar de objetos distribuidos creado por el consorcio OMG en el que se reúnen un importante grupo de más de 700 empresas en un intento por impulsar el desarrollo de objetos distribuidos y de la tecnología de agentes.

Las características del protocolo TAP indican que RMI puede acabar siendo insuficiente para soportar con garantías las comunicaciones que van a verse implicadas, por lo que CORBA puede ser una opción más interesante que será necesario evaluar como otra posibilidad más robusta para la implementación del prototipo de red final. Además, en el caso de CORBA nos encontraremos también con los aspectos de normalización de los SMA, de forma que podrán tenerse en cuenta también las propuestas del OMG en el desarrollo del subsistema SMA-TAP del prototipo propuesto.

La ampliación y mejora del SMA-TAP es, por tanto, otra de las futuras líneas de acción de modo, que puedan analizarse en detalle todas las posibilidades estándares propuestas por OMG y/o FIPA desde el punto de vista de la tecnología de agentes debidamente normalizadas.

Una vez se disponga de una plataforma de TAP en red e implementada según los estándares establecidos para los agentes podremos obtener datos más fiables de las posibilidades de TAP, aunque se verá afectado por los negativos efectos de rendimiento debidos a lenguajes de *script* y de una JVM implementada en software.

13.1.4. CONMUTADORES ACTMS Y ARQUITECTURAS ESPECIALIZADAS

La implementación final de TAP como protocolo explotable está en relación directa con los nodos activos AcTMs. Estos conmutadores activos no sólo incorporan tecnología software, sino que también se han propuesto equipados con el hardware necesario para que el protocolo pueda tener un adecuado rendimiento. Desde este punto de vista, podría decirse que la implementación definitiva de los nodos activos pertenece al ámbito de las arquitecturas especializadas para poder dar lugar a un modelo real de AcTMs donde se diseñen y especifiquen los diferentes aspectos hardware que han sido identificados en la arquitectura TAP. Determinados componentes de la arquitectura pueden identificarse como claramente implementables por hardware, es el caso del buffer, la DMTE y tablas de E/S. No obstante, otros aspectos menos claros como la técnica VC Merge propuesta, o la propia gestión de las colas de entrada con el algoritmo QPWFQ y toda la actividad de comunicación relacionada con el buffer y la DMTE, podrían ser también implementados por hardware en un intento por conseguir un mejor rendimiento de la red y reducir los retardos

Por otro lado, otra interesante propuesta dentro del ámbito de las arquitecturas especializadas es la posibilidad de implementar también en hardware la JVM para conseguir evitar los problemas de rendimiento del lenguaje Java en la implementación de protocolos que se han destacado en el *Capítulo 12*. La máquina

virtual de Java implementada en software aporta muchas ventajas al lenguaje; sin embargo, cuando se requiere de un rendimiento optimizado, las posibilidades y ventajas de su implementación mediante técnicas hardware son evidentes. La implementación hardware de la JVM puede ser un importante complemento para los conmutadores activos AcTMs que se han descrito en esta tesis.

Otro aspecto de interés es considerar que TAP puede también dar soporte a las arquitecturas *Differenciated Services* (*assured forwarding*), por lo que será necesario evaluar esta posibilidad como una de las evoluciones futuras de TAP.

13.2. CONCLUSIONES FINALES

Se ha presentado TAP como una arquitectura de comunicaciones para la tecnología ATM, que se apoya en el propio modelo arquitectónico ATM, al que se ha añadido un protocolo con características activas que permite que los nodos de la red, equipados con un sistema multiagente, sean capaces de ofrecer a las conexiones que lo requieran garantía de servicio. Así, se ha presentado un nuevo parámetro de calidad de servicio derivado de los parámetros generales de QoS que hemos denominado como GoS. La GoS puede ser empleada por un conjunto de fuentes de tráfico privilegiadas que van a disponer de un tratamiento especial en las situaciones de congestión de forma que, cuando éstas se producen, son resueltas localmente en la propia red por los conmutadores AcTMs que implementan la arquitectura y el protocolo TAP y aprovechando los periodos de inactividad de los enlaces.

La respuesta local a las congestiones permite evitar las retransmisiones extremo-extremo con lo que se consigue además optimizar el *goodput*, beneficiando no sólo a las conexiones privilegiadas, sino también al resto de la red. El mecanismo de recuperaciones se aplica sobre las PDU como unidad de transferencia, para lo que se han propuesto nuevas posibilidades en la tecnología ATM como la extensión de su capa AAL-5, dando lugar a EAAL-5 cuyo principal objetivo es el de aportar el servicio de GoS y también servicios ordenado y/o desordenado del tráfico servido a los protocolos que están situados en capas superiores. Así, TAP no sólo beneficia al tráfico ATM nativo, sino también a protocolos tan extendido como TCP que pueden ver mejorado su rendimiento con las posibilidades propuestas.

Se ha justificado la característica activa del protocolo apoyándonos en la especificación de un sistema de agentes múltiples que desempeñan una labor activa en la evitación, detección y resolución de las situaciones de congestión. Igualmente se ha argumentado el carácter distribuido de la arquitectura, de forma que se propone el uso de TAP en una VPN constituida por conmutadores AcTMs junto a otros que no soportan TAP. La intención es justificar la posibilidad de integración de TAP con la tecnología ya existente, ya que nuestra propuesta se basa en las recomendaciones estándares, lo que permite su coexistencia con todos ellas.

Se han realizado también adaptaciones sobre algoritmos de gestión justa de colas de entrada a los conmutadores, de donde ha surgido la propuesta QPWFQ para evitar que las fuentes privilegiadas puedan afectar negativamente a las que no lo son. Además, se ha propuesto una mejora sobre EPD para soportar el mecanismo de retransmisiones, dando lugar a EPDR, que se aplica sobre el buffer de los conmutadores activos para evitar las congestiones o resolverlas cuando son inevitables. Se ha explicado también la técnica VC-merge aplicada para evitar problemas de mezclas de flujos. En suma, se ha recurrido a la ingeniería de protocolos, enriquecida con el paradigma de agentes software, para lograr el objetivo general de las transferencias garantizadas. Pero además se consigue evitar otro tipo de problemas particulares como son la fragmentación de las PDU, la implosión sobre las fuentes de tráfico, el *interleaving* a la salida de los conmutadores y, además, se mejora el rendimiento general de toda la red.

Se ha presentado el modelo de arquitectura en el que se apoyan los nodos activos, destacando sus componentes hardware, así como el conjunto de algoritmos que constituyen el protocolo TAP y el subsistema de agentes software en los que se apoya. También se ha realizado un estudio de rendimiento de la propuesta a través de un prototipo que demuestra su viabilidad recuperando en la red activa un importante número de PDU que de otro modo se perderían y tendrían que ser recuperadas entre los extremos de la comunicación. Se ha demostrado también la viabilidad de aprovechar los tiempos de inactividad de la red para atender las congestiones. Del mismo modo, se han demostrado las ventajas del algoritmo QPWFQ en la gestión de las colas de entrada y en el mecanismo de retransmisiones.

Finalmente, se han identificado las líneas de acción futuras que permitan continuar la investigación propuesta en esta tesis para dotar a la actual tecnología de una solución robusta y completa por la que se pueda disponer de un Protocolo Activo para Transmisiones Garantizadas sobre una Arquitectura Distribuida y Multiagente en Redes ATM.

BIBLIOGRAFÍA(*)

[Abraham,98] Abraham, P., and Kumar, A., "A Stochastic Approximation Approach for Max-Min Fair Adaptive Rate Control of ABR Sessions with MCRs," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 3*, pp. 1358-1365, 1998.

[Adiseshu,96] Adiseshu, Parulkar, G., and Varghese, G., "A reliable and Scalable Striping Protocol," *SIGCOMM'96*, 1996.

[Almesberger,98] Almesberger, Ferrari, T., and Le Boudec, J.-Y., "SRP: a Scalable Resource Recervation Protocol for the Internet," *Technical Report SSC/1998/009, http://sscwww.epfl.ch*, 1998.

[Armitage,96] Armitage, G. J., "Multicast and Multiprotocol support for ATM based Internets," *ACM SIGCOMM Computer Communications Review*, pp. 34-46 1.996.

[Armitage,97] Armitage, G. J., "IP Multicasting over ATM Networks", *IEEE Journal on Selected Areas in Communications*, Vol 15, No. 3, pp. 445-457, 1997.

[Benech,97] D. Benech, T. Desprats, Y. Raynaud. "A KQML-CORBA based Architecture for Intelligent Agents Communication in Cooperative Service and Network Management", 1997.

[Bieszxzad,98] A. Bieszczad, B. Pagurek, T. White. "Mobile Agents for Network Management". IEEE Communications Surveys. 1.998. http://www.comsoc.org/pubs/surveys/4q98issue/bies.html>

[Borger,97] Borger L., "RSVP over ATM Implementation Guidelines", *Internet Draft*, June 1997.

[Campbell,94] Campbell, A., Coulson, G., and Hutchison, D., "A Quality of Service Architecture", *ACM Computer Communications Review*, April 1994.

[Campbell,97] Campbell, A. T., and Coulson, G., "QoS Adaptive Transports: Delivering Scalable Media to the Desktop", *IEEE Network*, pp. 18-27, March/April 1997.

[Chess,95] D. Chess, C. Harrison, A. Kershenbaum. "Mobile Agents: Are They a Good Idea?" . I.B.M., 1.995.

[Conrad,97] Conrad, P., Amer, P., Golden, E., Iren, S., Marasli, R., and Caro, A., "Transport QoS over Unreliable Networks: No Guarantees, No Free Lunch!," *IWQOS'97*, pp. 315-318, 1997.

[Dabbous,97] Dabbous, "High performance Protocol Architecture," *Computer Networks and ISDN Systems, 29, pp.* 735-744, 1.997.

[Delgross,95] Delgross, L., Berger, L.,"Internet Stream Protocol Version 2 (ST2) Protocol Specification-Version ST2+," Aug. 1995.

[Djemame,97] K. Djemame, and M. Kara, "Proposals for a Coherent Approach to Cooperation between TCP and ATM Congestion Control Algorithms," 1997.

[Falchuck,98] B. Falchuck, A. Karmouch. "Visual Modelling for Agent-Based Applications". Computer, diciembre 1.998.

[Gai94] D. Gaiti, "Distributed Artificila Intelligence as an AIP architecture for Network Management", Elsevier Science Publishers B.V., Advanced Information Processing Techniques for LAN and WAN Management C-17, pp.261-272, 1994.

[Gauthier,95] Gauthier, and Le Boudec, J.-Y., "Scalability Enhancement for Connection-Oriented Networks," *http://sscwww.epfl.ch*, 1995.

[Giordano,97] Giordano, Le Boudec, J.-Y., Oechslin, P., and Robert, S., "VBR over VBR: the Homogeneous, Loss-free Case," *INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Driving the Information Revolution., Proceedings IEEE Vol. 1*, pp. 168-176, 1997.

[Giordano,97] Giordano, Schmid, R., Beeler,R., Flink, H.,and Le Boudec, J.-Y., "IP and ATM - a position paper," *Technical Report SSC/1997/017, http://sscwww.epfl.ch*, 1997.

[Goyal,96] Goyal, P., Vin, H., Shen, C., Shenoy, P., "A reliable, Adaptive Network Protocol for Video Transport",

(*) Esta bibliografía ha sido también empleada de forma significativa, aunque no haya sido referenciada en los capítulos de la memoria de tesis.

INFOCOM'96, Vol 3, pp. 1080-1090, 1996.

[Gustafsson,97] Gustafsson, E., and Karlsson, G., "A Literature Survey on Traffic Dispersion", *IEEE Network*, pp. 28-36, March/April 1997.

[Hayzelden,98] A. L. G. Hayzelden and J. Bigham, "Heterogeneous Multi-Agent Architecture for ATM Virtual Path Network Resource Configuration," 1998.

[Handël,95] Handël, R., Huber, M. N., Schoröder, S., "ATM Networks Concepts, Protocols. Applications (2ª Ed.)," *Addison-Wesley*, 1995.

[Hong,98] Hong, and T. Suda, "Performance of ERICA and QFC for Transporting Bursty TCP Sources with Bursty Interfering Traffic," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 3 pp.1341-1349*, 1998.

[Huard,96] Huard, J.-F., "kStack: A user space native-mode ATM transport layer with QoS support," *Technical Report CU/CTR TR 463-96-29*, Center for Telecommunications Research, Columbia University, New York, (1996) *http://comet.ctr.columbia.edu/software/ksatck*

[IKV,98] "Grasshopper: The Agent Platform. Technical Overview". IKV++ GmbH. 1.998.

[Kai,95] Kai-Yeung Siu, Hong-Yi Tzeng, "Congestion control for multicast service in ATM networks,", IEEE GLOBECOM'95, pp. 310-314, (1995).

[Karjoth,97] G. Karjoth, D. B. Lange, M. Oshima. "A Security Model for Aglets". IEEE Internet Computing, julio-agosto 1.997.

[Koifman,96] Koifman, and Zabele, S., "RAMP: A Reliable Adaptive Multicast Protocol," *INFOCOM'96, Vol. 3 pp*. 1442-1451, 1996.

[Liu,97] Liu, X., and H. T. Mouftah, "Queuing Performance of Copy Networks With Dynamic Cell Splitting for Multicast ATM Switching", *IEEE Transactions on communications*, vol. 45, No. 4, April, 1997.

[Nikolaos,98] Nikolaos Anerousis, Aurel A. Lazar, "An architecture for Controlling Service Demand in ATM Networks based on Pricing Agents," 1.998.

[Nonnenmacher,98] Nonnenmacher, and Biersack, E., "Optimal Multicast, Feedback,"*INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 3* pp. 964-971, 1998.

[Papadopoulos,98] Papadopoulos, C., Parulkar, G., and Varghese, G., "An error Control Scheme for Large-Scale Multicast Applications," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 3*, pp. 1188-1196, 1988.

[Partridge,95] Partridge, Hughes, J., and Stone, J., "Performance of Checksums and CRC over Real Data," *SIGCOMM'95, pp. 68-76*, 1995.

[Pitsillides,96] Pitsillides, A., Ioannou, P., and Tipper D., "Integrated Control of Connection, Flow Rate, and Bandwidth for ATM based Networks", INFOCOM'96, vol 2, pp.785-793, 1995.

[Pitts,97] J. M. Pitss, "Introduction to ATM. Design and Performance", Ed. Wiley, 1997.

[Pricker,95] M. de Prycker, "Asynchronous Transfer Mode. Solution for Broadband ISDN (3ª Ed.)," *Ed. Prentice Hall.*,(1995).

[Raha,96] Raha, A., Kamat, S., and Zhao, W., "Admission Control for Hard Real-Time Connections in ATM LANs", *INFOCOM 1996*, Vol. 1, pp. 180-188, 1996.

[Ravindran,95] Ravindran K., "A Flexible Network Architecture for Data Multicasting in "Multiservice Networks"," *IEEE Journal on Selected Areas in Communications*, vol. 13, No 8, pp.1426-1444, Oct. 1995.

[Rizzo,97] Rizzo, L., "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review* pp. 31-41, 1997.

[Rosenberg,98] Rosenberg, J., and Schulzrinne, H., "Timer Reconsideration for Enhanced RTP Scalability," *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE Vol. 1*, pp. 233-241, 1998.

[Salama,97] Salama, H., Reeves, D., and Viniotis, Y., "Evaluation of Multicast Routing Algorithms for Real-Time Communications on High-Speed Networks", *IEEE Journal on Selected Areas in Comm*. Vol. 15 No. 3, April 1997.

[Shacham,97] Shacham and Yokota, H., "Admission Control Algorithms for Multicast Sessions with Multiple streams," *IEEE Journal on Selected Areas in Communications Vol 15*, Nº 3, pp. 557-566, April 1.997.

[Simmons,97] Simmons, J., "Proof of correctness of ATM retransmission scheme," *Computer Network and ISDN Systems* 29, pp. 181-194, 1997.

[Sisalem,97] Sisalem, D., "End-To-End Quality of Service Control Using Adaptive Applications," *IWQOS'97*, pp. 379-390, 1997.

[Strayer,92] Strayer, T., Dempsey, B. J., and Weaver, A. V., "XTP - The Xpress Transfer Protocol," *Addison-Wesley Publishing Company*, (July 1992).

[Veitch,97] Veitch, P., and Johnson, D., "ATM Network Resilience", *IEEE Network*, pp. 26-33, Sept/Oct. 1997.

[Venners,97] B. Venners. "Solve Real Problems with Aglets, a Type of Mobile Agents". JavaWorld, mayo 1.997. http://www.javaworld.com/javaworld/jw-05-1997/jw-05-hood.html

[Visi,98] "VisiBroker Version 3.3 Programmer's Guide. Naming and Event Services". Inprise Corp. 1.998.

[Yang,96] Yang and J. Huang, "A multimedia Synchronization model and Its implementation in Transport Protocols," *IEEE Journal Selected and Communications Vol. 14 Nº 1, pp. 212-225*, Jan. 1996.

[Widjaja,96] I. Widjaja, M. F. Neuts, and Li Jian-Min, "Conditional overflow probability and profile curve for ATM congestion detection," *IEEE Proceedings INFOCOM'96*, pp. 970-977, (1996).

Rec. I.321 Protocol Reference Model.

Rec. I.150 B-ISDN ATM functional char ITU-TS 1.993.

Rec. I.356. "B-ISDN ATM Layer Cell Transfer Performance," *Frozen issue* March 1993.

Rec. I.361, "Especificación de la capa modo de transferencia asíncrono de la RDSI-BA," *UIT-T*, (11/1995).

Rec. I.362 ATM Adaptation Layer.

Rec. I.363, "Especificación de la capa de adaptación del modo de transferencia asíncrono de la RDSI-BA," *UIT-T*, (03/1993).

Rec. I.363.1, "Capa de adaptación del modo de transferencia asíncrono tipo 1," *UIT-T*, (08/1996).

Rec. I.363.3, "Capa de adaptación del modo de transferencia asíncrono tipo 3/4," *UIT-T*, (08/1996).

Rec. I.150, "B-ISDN ATM functional characteristics", *ITU-TS*, (1993).

____ "Native ATM Service: Semantic Description Version 1", ATM Forum Technical Committee", *ATM Forum Document af-saa-0048.000*, (Feb 1996).

____ "ATM User-Network Interface (UNI) Signalling Specification Version 4.0," *ATM Forum Technical Committee, ATM Forum Document af-sig-0061*, (July 1996).

____ "ATM User Network Interface Specification V3.0", *Prentice-Hall, Inc.*, pp. 189-193, (1993).

____ "ATM user-network interface version 3.1 specification", *ATM Forum*, (1994).

____ "ATM Name System Specification", Version 1.0, *ATM Forum*, (Nov. 1996).

GLOSARIO DE TÉRMINOS E ÍNDICE TEMÁTICO