Protocolo Activo para Transmisiones Garantizadas sobre una Arquitectura Distribuida y Multiagente en Redes ATM[1]

José Luis González-Sánchez (*) y Jordi Domingo-Pascual (+)

(*) Universidad de Extremadura.
Escuela Politécnica de Cáceres. Avda. Universidad S/N. (10.071) Cáceres
E-Mail: jlgs@unex.es

(+) Universitat Politécnica de Catalunya.
Campus Nord, Modul D6. Jordi Girona 1-3 (08.034) Barcelona
E-Mail: jordi.domingo@ac.upc.es

Sinopsis

Presentamos TAP (*Trusted and Active 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 VPIs/VCIs. Se propone también una extensión de la capa AAL-5 de ATM que hemos denominado *Extended AAL type 5* (EAAL-5) usada para la gestión de las conexiones privilegiadas extremo-extremo. 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. El protocolo propuesto emplea mecanismos *NACK* (mediante células RM de retorno) y es soportado por conmutadores ATM activos que hemos equipado con una memoria de almacenamiento de PDUs denominada *Dynamic Memory to store Trusted native EAAL-5 PDUs* (DMTE). La arquitectura activa está basada en un sistema de multiagentes programables colaborativos y distribuidos en la red. Varias simulaciones demuestran la efectividad del mecanismo propuesto con un mejor *goodput* en la red. Las simulaciones a través de fuentes ON/OFF analiza conexiones punto-a-punto y punto-a-multipunto usando objetos, *threads*, sincronizaciones y procesos distribuidos implementados en lenguaje Java.

Palabras clave: Redes ATM, protocolo activo distribuido, arquitectura distribuida, sistema multiagente, redes activas, transferencias garantizadas.

1. Introducción e investigaciones relacionadas

La tecnología ATM se caracteriza por su excelente comportamiento ante diversos tipos de tráfico y por ofrecer la posibilidad de negociar los parámetros de *Quality of Service* (QoS) [1] como el *throughput, delay, jitter* y la *reliability*. 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 Check* (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. El principal problema de esto es que la pérdida de una única célula provoca errores de reensamblado CRC en la capa AAL-5, lo que requiere la retransmisión de una PDU completa.
Las redes ATM experimentan tres tipos de errores [2-5]: pérdida de células debidas a la congestión de los conmutadores; corrupción parcial de datos causadas por bits erróneos y errores de conmutación debidos a

[1] Este trabajo ha sido patrocinado en parte por la CICYT Proyecto No. TEL99-1117-C03-03.

corrupciones no detectadas de las cabeceras de las células. Destacamos que las congestiones son el tipo de errores más habitual y es aquí donde situamos nuestro trabajo para ofrecer las transferencias garantizadas con TAP.
La literatura describe tres técnicas básicas para ofrecer fiabilidad: *Automatic Repeat Request* ARQ [6], *Forward Error Correction* FEC [7-11] y mecanismos híbridos de ARQ combinados con FEC.
Mientras ARQ añade latencia (debida al coste de los NACKs) e *implosion* (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 *implosion*. Las transmisiones multicast no pueden basarse en retransmisiones desde la fuente. En TAP, los nodos activos intermedios se encargan de las retransmisiones.
Los esquemas de control de congestión más comúnmente usados para ofrecer *throughput* y justicia, mientras se minimiza el retardo en las redes ATM son: *Random Cell Discard* (RCD), *Partial Packet Discard* (PPD*), Early Packet Discard* (EPD) [12]*; Early Selective Packet Discard* (ESPD), *Fair Buffer Allocation* (FBA) y *Random Early Detection* (RED). TAP usa EPD para aliviar el efecto de las congestiones y de la fragmentación de paquetes.
ATM Adaptation Layer type 5 (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 servicio garantizado con retransmisiones y es también compatible con AAL-5 nativo. En este trabajo proponemos un mecanismo para aprovechar los periodos de inactividad (*idle*) de las fuentes para retransmitir *las Common Part Convergence Sublayer PDU* (CPCS-PDU) de EAAL-5.
Las redes activas, abiertas y programables son una nueva área técnica [13-17] 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 protocolos moviendo 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. Una red es activa si hay nodos activos en sus árboles de distribución con la capacidad de ejecutar programas de usuario, y también si implementan mecanismos de propagación de código. 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 las propuestas son insuficientes para redes ATM.
Se han aportado características activas a TAP a través de mecanismos hardware y técnicas software. La arquitectura TAP incluye el agente *Control Congestion Agent* (CCA) para gestionar las retransmisiones solicitadas entre parejas de nodos activos; el agente *Dynamic Memory to store Trusted native EAAL-5 PDUs Agent* (DMA*)* para recuperar PDUs desde la memoria DMTE; *el Dispacher Agent* (DPA) procesa las PDUs desde la DMTE a los puertos de salida de los conmutadores activos. El agente *Class of Service Agent* (CSA) se encarga de caracterizar las fuentes de tráfico.
Hemos simulado y estudiado transferencias combinadas con conmutadores activos y otros no activos para constituir una *Virtual Private Network* (VPN*). Java Development Kit V1.2.1* ha sido el lenguaje y entorno usado para implementar TAP por las especiales características ofrecidas por este lenguaje.
La sección 2 describe la arquitectura TAP. En las secciones 3 y 4 presentamos el prototipo de conmutador activo para soportar la arquitectura. La sección 5 describe el rendimiento y orienta nuestro trabajo para la mejora del protocolo TAP. Finalmente, ofrecemos las conclusiones en la sección 6.

2. Descripción general de la arquitectura TAP

AAL-5 fue propuesto [4] para reducir las sobrecargas de cabeceras introducidas por AAL3/4. La Figura 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 PDUs tiene 4 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.

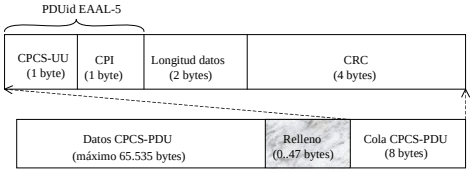

Figura 1 Formato de CPCS-PDU EAAL-5

TAP usa dos octetos como número de secuencia, que es asignada extremo-extremo por los usuarios EAAL-5 para evitar el recálculo de CRCs y la modificación de la cola de CPCS-PDUs. El CRC es empleado como en AAL-5 para detectar los bits erróneos en la CPCS-PDU.
Para implementar NACK usamos células estándares *Resource Management* (RM), sin frecuencia fija y generadas cuando se congestiona un conmutador. Esto se usa para aliviar el efecto negativo de las sobrecargas de cabeceras debido a que un número fijo de células RM desaprovecharía el ancho de banda.
Cuando se produce congestión en un conmutador, EPD desecha la PDU que se está transmitiendo en el momento de la congestión. Luego el agente DMA busca la PDU en la DMTE. Si esta PDU no se encuentra en la DMTE local, el agente CCA del nodo activo genera una célula RM que es transmitida en sentido contrario al flujo de información indicando el número de secuencia de la PDU buscada. La célula RM también debe contener el VPI/VCI para identificar la conexión que ha experimentado los problemas. Este mecanismo requiere relacionar las fuentes de tráfico con sus puertos de E/S para aliviar el efecto de valores iguales de VPI/VCI en puertos diferentes. Los octetos 22 a 51 de las células RM almacenan el identificador *Port/VPI/VCI/PDUid* de las PDUs solicitadas (ver Tabla 1).

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

Tabla 1 Campos y sus posiciones en las células RM [18]

Cuando la célula RM llega a un conmutador activo, TAP busca la PDU solicitada y, si aún está en DMTE, la PDU es retransmitida si el *idle time* de la conexión es suficiente.
Al llegar un NACK (célula RM) a un conmutador no activo éste procesa sólo la célula RM y la reenvía a su conmutador vecino en la dirección contraria al flujo de datos. Los conmutadores no activos no tienen DMTE para la recuperación de PDUs y su función es sólo el envío (o reenvío) de PDUs hasta el destino.
Para concluir este apartado enfatizamos que TAP no puede ofrecer fiabilidad completa, pero asegura y recupera un importante número de PDUs que, en otro caso, se perderían por congestión. El mecanismo también garantiza que no hay retransmisiones extremo-extremo y sí entre pares de nodos activos. El mecanismo de retransmisiones genera PDUs desordenadas en los receptores. El protocolo ofrece dos tipos de servicio. El primero llamado SEQ (secuencial) ordena las PDUs y, cuando detecta un fallo de secuencia, asume que la PDU se ha perdido y deja su recuperación a protocolos de capas superiores (TCP). El segundo tipo de servicio es el desordenado (*connectionless*) que no hace ningún tipo de ordenación. Estos dos servicios son ofrecidos por la propuesta EAAL-5.
La Figura 2 presenta la arquitectura TAP incluyendo la memoria DMTE y los agentes. 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 segundo conmutador activo.

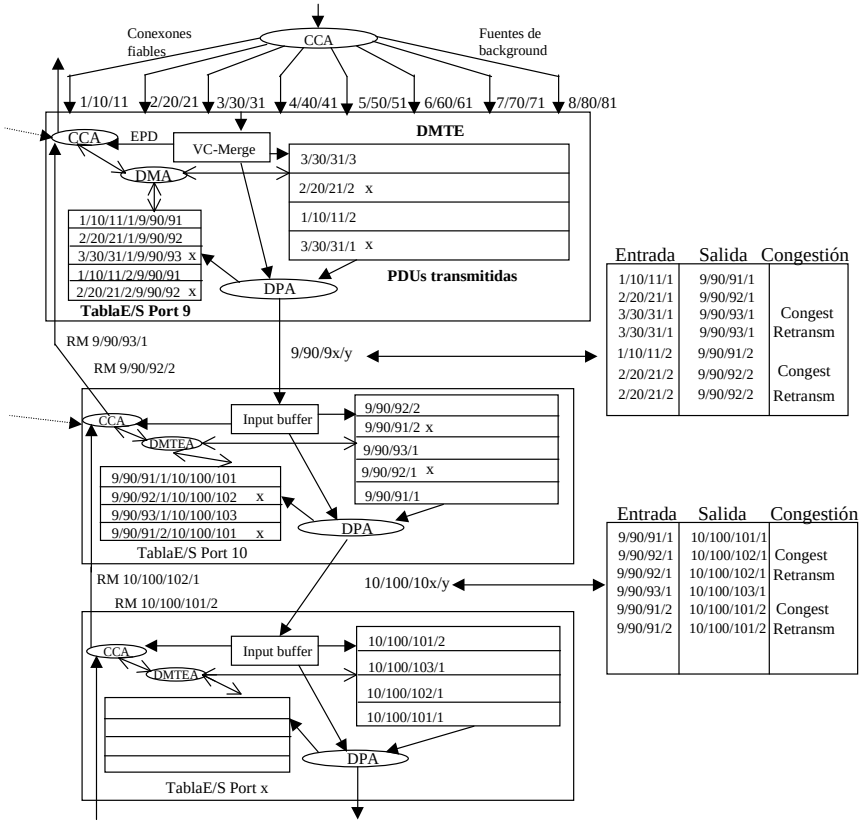

Figura 2 Arquitectura TAP

3. Memoria DMTE (Dynamic Memory of Trusted EAAL-5 PDUs)

La arquitectura del conmutador ATM es similar a un conmutador con buffers de salida con características *VC-merge* y la DMTE es el módulo clave. La principal función de esta memoria es el almacenamiento temporal de PDUs en el nodo activo después que han sido transmitidas al puerto de salida, así podrán ser solicitadas en las retransmisiones. El protocolo TAP hace una copia de cada PDU que llega al nodo activo (mientras se realiza el VC-merge). Se almacenan varias PDUs por cada conexión privilegiada. Cuando una PDU completa llega a la DMTE esta es "copiada" completamente en el buffer de salida. La PDU permanece en la DMTE hasta que es necesario el espacio para almacenar una nueva PDU.

Debido al gran tamaño que puede tener una PDU (hasta 65.535 bytes), y por el potencialmente elevado número de conexiones (VPI/VCI), el tamaño de la DMTE puede ser excesivo. Esta es la razón por la que el número de PDUs almacenadas para cada conexión es limitado y también el número de VCIs con transferencias garantizadas.

TAP accede a la DMTE a través de un índice consistente en el número de port, el identificador de PDUid (que corresponde a los campos UU y CPI de AAL5) y los valores VPI/VCI.

Mientras una PDU procedente de la DMTE está siendo retransmitida, si existe una PDU entrante con el mismo VPI/VCI, y no existe espacio libre en la DMTE la PDU entrante es desechada. Esto es similar a la pérdida causada por la falta de *buffers VC-merge*.

La estimación del tamaño de memoria necesario para el almacenamiento de las PDUs demuestra que, como puede verse en los cálculos presentados a continuación, no es necesario un coste elevado en hardware para el soporte de la memoria dinámica DMTE. El tamaño de la memoria depende, lógicamente, de los tamaños de las PDUs procesadas, por lo que planteamos los cálculos en función de la situación más desfavorable (PDUs del tamaño máximo de 64 Kbytes) y de la más factible y probable en el caso de la transferencia de segmentos TCP de 1.500 octetos. En ambas casos suponemos almacenar 3 PDUs por cada una de las conexiones confiables:

En el caso de PDUs EAAL-5 de tamaño máximo tendremos:

3 PDUs * 64 Kb cada PDU= 192.000 Kb * 10 conexiones confiables = 1,9 Mb.
3 PDUs * 64 Kb cada PDU= 192.000 Kb * 20 conexiones confiables = 3,8 Mb.

Si consideramos PDUs de tamaño 1.500 bytes obtendremos:

3 PDUs * 1,5 Kb cada PDU= 4,5 Kb * 10 conexiones confiables = 45 Kb.
3 PDUs * 1,5 Kb cada PDU= 4,5 Kb * 100 conexiones confiables = 450 Kb.
3 PDUs * 1,5 Kb cada PDU= 4,5 Kb * 1.000 conexiones confiables = 4,5 Mb.

Podemos observar que con 4,5 Mbytes de memoria, TAP es capaz de garantizar 1.000 fuentes TCP.

4. Sistema multiagente y distribuido

Todas las redes tienen 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 (*active-nets*) 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 estudia varios mecanismos para aprovechar los nodos activos. Sin embargo, las propuestas son insuficientes para las redes de tecnología ATM [13-21], y la referencia [14] es un ejemplo de estas recientes investigaciones para ATM.

No hay consenso para decidir cuándo una red es activa y 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 conmutadores ATM. 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.

El conmutador ATM de nuestro modelo actúa como un buffer de salida que lee la información de VPI/VCI de las células que llegan para ser reenviadas a su correspondiente puerto de salida. Pero equipamos este conmutador con técnicas hardware y software activas para alcanzar nuestros objetivos. TAP usa cuatro agentes para realizar las siguientes funciones. El agente CCA controla las congestiones basándose en EPD. Este agente monitoriza el buffer de entrada y cuando su ocupación sobrepasa el umbral establecido no acepta ninguna PDU entrante. La última célula EAAL-5 contiene el VPI y VCI de la cabecera y el PDUid en la cola de la EAAL5-CPCS-PDU. La PDU completa es desechada como en EPD pero la información sobre el Port, VPI, VCI y PDUid es usada para generar la solicitud de retransmisión de esa PDU. Si la PDU solicitada está todavía en la DMTE local ésta puede ser recuperada y reenviada al buffer de salida. En otro caso, la solicitud debe ser reenviada por el agente CCA al nodo activo previo en el sentido contrario del flujo.

El agente DMA se coordina con el agente CCA para solicitar la PDU al conmutador activo anterior. La función del agente CCA 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 en el sentido del emisor.

Cuando un agente CCA recibe una célula RM busca la PDU en la memoria DMTE a través del agente DMA usando el valor */Port/VPI/VCI/PDUid* como índice. Si la PDU está todavía, se comprueba que el flujo de células está en estado *idle* y la PDU puede se recuperada. Destacamos que los valores de PDUids son asignadas extremo-a-extremo para cada VCC y no hay ningún cambio por interpretación de la PDU solicitada. Actualmente trabajamos en el uso de células RM como mecanismo de transporte para llevar la propagación de código entre nodos activos. Este código contiene instrucciones para optimizar la retransmisión de PDUs en conexiones multipunto. El agente CCA usa estas instrucciones para examinar el árbol de distribución en anchura ofreciendo mejor *goodput* en las retransmisiones. Otro importante aspecto es que los CCAs no son un protocolo en el sentido clásico, es decir, son sólo una oportunidad para recuperar una PDU. No se usan ni ventanas ni *timeouts* en las retransmisiones de nuestra propuesta.

El agente DPA se encarga de obtener las PDUs del buffer de entrada o de la DMTE y despacharlas al correspondiente puerto de salida, previa actualización de la Tabla de Entrada/Salida de ese puerto. CSA, como CCA, es un agente programable que se encarga de la caracterización del tráfico entrante.
El uso de sistemas multiagente en comunicaciones aportan importantes beneficios [27] sobre otras tecnologías existentes pueden resumirse en los siguientes:
Los sistemas multiagente son protocolos del nivel de aplicación situados sobre capas distribuidas como CORBA que ofrecen un nivel de aplicación común a todos los agentes del sistema y también protocolos de interacción que facilitan la colaboración entre agentes y la negociación de sus actividades. Por otro lado, el sistema multiagente ofrece determinados niveles de escalabilidad que permiten el uso de nuevas autoridades y la expansión de las capacidades de las ya existentes. Por último es también destacable que la gestión distribuida permite reducir los retardos en las conexiones entre los conmutadores lo que repercute directamente en la mejora de los parámetros de QoS de redes como ATM.
Además, la coordinación entre los nodos activos que proponemos en nuestro sistema multiagente distribuido aporta interesantes ventajas específicas en el control de tráfico de las redes ATM como, por ejemplo, los agentes pueden optimizar el uso de los recursos disponibles. La naturaleza adaptativa y distribuida de los agentes nos permite tener una arquitectura más flexible y escalable. También se facilitan las labores de intervención humana en el control de la red, y la toma de decisiones de los agentes puede facilitar la integración de conmutadores de diferentes fabricantes. Por otro lado, las decisiones de conexión permiten una mejor caracterización de las fuentes de tráfico y de sus parámetros de QoS. Por último, podemos tener redes más fiables pues los agentes aportan una gestión distribuida que TAP aprovecha especialmente.

5. Evaluación de rendimiento

En trabajos previos [22, 26] hemos presentado y demostrado el buen comportamiento del protocolo RAP sobre la arquitectura TAP. Hemos simulado varias técnicas software para introducir características activas en los conmutadores. Estos mecanismos controlan y gestionan los VCIs privilegiados y también ofrecemos un mecanismo activo para recuperar PDUs de los conmutadores activos vecinos y buscando caminos óptimos cuando se retransmite una PDU. La simulación nos permite definir la probabilidad de congestión en los emisores, receptores y en los conmutadores ATM. Cuando un nodo está congestionado éste solicita la retransmisión de la correspondiente PDU. La simulación también permite al usuario introducir valores variables como los parámetros de las fuentes ON/OFF, el número de receptores o el número de conmutadores activos y no activos.

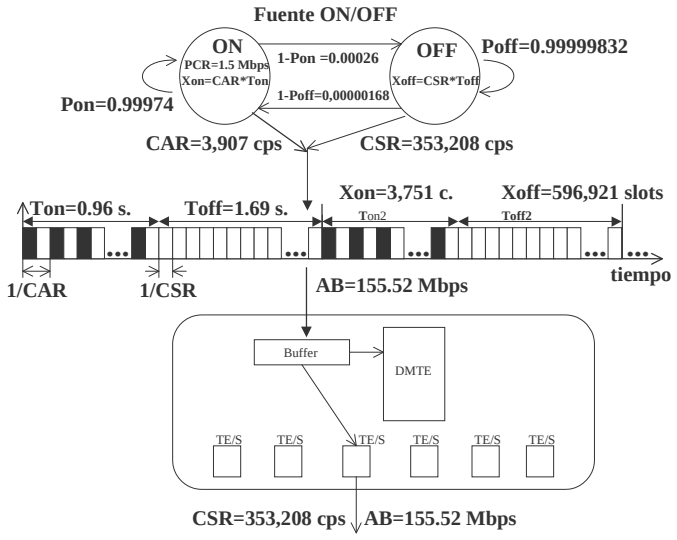

Figura 3 Patrón de células para una fuente ON/OFF

En la simulación se usan fuentes de tráfico ON/OFF (ráfagas) para analizar el comportamiento de la pérdida de células. El modelo ON/OFF [23,24] se usa para caracterizar el tráfico ATM por conexiones

unidireccionales. La Figura 3 muestra este modelo como una fuente que envía (estado ON) datos CSCP-PDU-EAAL-5 durante un tiempo t_{on} a una velocidad R o PCR (*Peak Cell Rate*) o que permanece inactiva o en silencio (estado OFF) sin producir células durante un tiempo t_{off}
La Tabla 2 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 PDUs de 1.500 octetos. Durante los estados ON estos procesos generan células a una velocidad de llegada R o PCR.

Descriptor de las fuentes de tráfico	**Parámetros**	**Mínimo**	**Máximo**
Ancho de Banda	BS	64 Kbit/s.	25 Mbit./s.
Velocidad de llegada de células	R o PCR	167 células/s.	65,105 células/s.
Tiempo entre células	1/R	6 ms.	15 μs.
Ancho de Banda del enlace	BL	155.52 Mb/s.	622 Mbit/s.
Velocidad de ranuras de células	C o CSR	353,208 células/s.	1,412,648 células/s.
Tiempo de servicio por célula	1/C	2.83 μs.	0.70 μs.
Periodo de tiempo activo	t_{on}	0.96 s.	1 s.
Media de células en un estado activo	X_{on}	160 células	65,105 células
Tiempo en estado de silencio	t_{off}	1.69 s.	2 s.
Media de ranuras vacias en un estado de silencio	X_{off}	596,921 ranuras de células	2,825,296 ranuras de células

Tabla 2 Descriptores de tráfico fuentes ON/OFF

La fuente genera ranuras de tiempo vacías. Usamos en todos los ejemplos una *C* o *CSR (Cell Slot Rate*) de C=353.208 células/s con un modelo de red con enlaces de 155,52 Mbit/s. Cuando la velocidad de llegada de células es menor que C, hay ranuras de tiempo vacías durante el estado activo (ver Figura 3).
Estudios empíricos [23] demuestran que $t_{on} = 0.96$ s. y $t_{off} = 1.69$ s. y usamos estos valores en las simulaciones, aunque también hemos usado otros valores para analizar sus efectos sobre TAP. Destacamos que hemos variado algunos de los parámetros para analizar el comportamiento de TAP cuando varía el escenario y los descriptores de las fuentes de tráfico, según mostramos en esta sección.
En la simulación de *N* fuentes idénticas que operan independientemente sobre un conmutador activo con un buffer de entrada de tamaño *X* y de *C* células/segundo de capacidad, con tiempos *Ton* y *Toff* de duraciones medias de tiempo de actividad y silencio respectivamente, calculamos el *Mean Cell Rate* (*MCR*),

$$MCR = PCR\ (Ton / (Ton + Toff)$$

Por otro lado, el *Factor de Actividad* (*FA*) del escenario de las *N* fuentes ON/OFF es,

$$FA = MCR / PCR = Ton / (Ton + Toff)$$

Podemos calcular el número de veces (*F* fuentes) que el *PCR* tiene cabida en la capacidad de servicio *C*,

$$F=C / PCR,$$

lo que nos permite afirmar que, en el ejemplo mostrado en la Figura 3, *F* = 353.208/3.751 = 94,1 fuentes ON/OFF confiables de *PCR* = 1,5 Mbps multiplexadas sobre un enlace de 155,52 Mbps de ancho de banda. En este caso, tendremos suficiente tiempo de OFF agregado que nos permita la recuperación de PDUs, siempre que no superemos las 94 fuentes multiplexadas.
La configuración básica (punto-a-punto) estudiada consiste en 3 conmutadores ATM activos. La Figura 4 muestra el resultado de variar el PCR entre 60 y 2.000 células por segundo. Como podemos ver, cuando la velocidad de llegada es baja, el número de PDUs recuperadas incrementa. Se observa también que el número de NACKs no enviados (PDUs no retransmitidas) es mayor cuando el valor de PCR incrementa. En este caso, la red no es sobrecargada con retransmisiones innecesarias. En esta simulación hemos fijado la fuente de datos que transmite 750 Kbytes; probabilidad de congestión = 10^{-3}. Para PCR=167 células/s.; t_{on}=0,96 s.; y t_{off}=1,69 s. y un total de 56 PDUs son desechadas por congestión; TAP recupera 11 PDUs, y 17 PDUs no son solicitadas. Cuando PCR alcanza 60 cells/s. el rendimiento es optimizado (27 PDUs recuperadas de 28) porque todas las PDUs perdidas son recuperadas y no se producen fallos de memoria DMTE (todas las PDUs solicitadas se encuentran en DMTE).

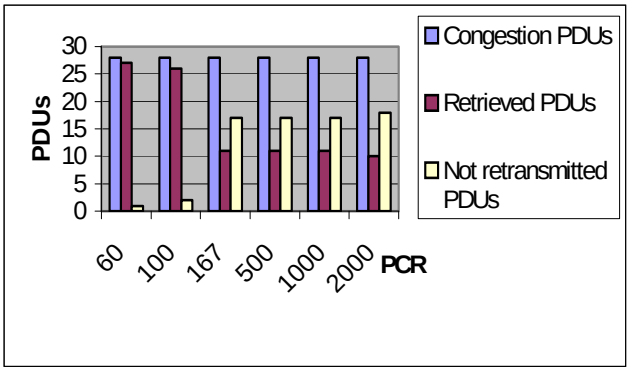

Figura 4 Efecto de la variación del PCR

Otro escenario consiste 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. Sólo varía el retardo en las transmisiones debido a los tiempos de propagación, pero el índice de PDUs recuperadas se mantiene como hemos demostrado anteriormente.

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. Actualmente trabajamos para conseguir conexiones multipunto-multipunto sobre TAP. Si consideramos los resultados anteriores podemos ver que intuitivamente el retardo total variará. También el tamaño de la memoria DMTE debe ser incrementado en los nodos activos para gestionar los VPI/VCI de las *n* conexiones privilegiadas.

No obstante, a continuación queremos describir varios aspectos en los que estamos trabajando para conseguir mejor *goodput*. Estamos considerando otros descriptores de las fuentes de tráfico como son *Sustainable Cell Rate* (SCR) y *Maximum Burst Size* (MBS). Con estos parámetros podremos caracterizar mejor el tráfico. También, como muchas aplicaciones usan el protocolo TCP para la transmisión de datos basadas en tramas, estamos trabajando para implementar la clase de *servicio Guaranteed Frame Rate* (GFR) [25] para ofrecer una mínima garantía de servicio a las clases de servicio UBR, VBR y ABR. Para soportar GFR simularemos fuentes con una garantía de *Minimum Cell Rate* (MCR) para un MBS y *Maximum Frame Size* (MFS). TAP ofrecerá garantía con la clase de servicio GFR que es capaz de distinguir tramas elegibles y no elegibles y también desecha células adecuadamente. Estamos también trabajando para ampliar las características activas de la arquitectura incluyendo otros agentes inteligentes para caracterizar el tráfico y sus clases de servicio.

6. Sumario

Este trabajo presenta TAP como una arquitectura distribuida para un protocolo activo que aprovecha las ventajas de los conmutadores activos propuestos. TAP gestiona un conjunto de VCIs privilegiados para ofrecer conexiones garantizadas cuando los conmutadores están congestionados. Para alcanzar estas características activas usamos un conmutador ATM activo con DMTE, una memoria dinámica que almacena PDUs de cada VCI privilegiado . Hemos verificado que es posible recuperar un importante número de PDUs sólo con DMTE y una razonable complejidad añadida en los conmutadores activos soportada por agentes software. El mecanismo de retransmisión está basado en ARQ con NACKs que genera células RM para solicitar las PDUs. Nuestras simulaciones demuestran que la idea intuitiva de aprovechar los tiempos de silencio en las fuentes ON/OFF es cierta. Así conseguimos también mejor comportamiento y QoS en las redes ATM. Estamos trabajando también para simular conexiones multipunto-multipunto para analizar el comportamiento de TAP en todo tipo de escenarios.

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] Recommendation I.361, "B-ISDN ATM Layer Specification," *ITU-T*, 11/1995.

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

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

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] D. L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine*, pp.80-86, January 1997.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] The ATM Forum Technical Committee. Traffic Management Specification Version 4.0 af-tm-0056.000, April 1996.

[19] German Goldszmidt, and Yechiam Yemini, "Delegated Agents for Network Management," *IEEE Communications Magazine*, Volume 36, 3, pp. 66-70, March 1998.

[20] G. Goldszmidt, and Y. Yemini, "Distributed Management by Delegation," *IEEE*, pp. 333-340, 1995.

[21] 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.

[22] José Luis González-Sánchez and Jordi Domingo-Pascual, "RAP: Protocol for Reliable Transfers in ATM Networks with Active Switches," *Technical report http://www.ac.upc.es/recerca/reports/INDEX1999DAC.html*, November 1.999.

[23] J. M. Pitss, "Introduction to ATM. Design and Performance," *Ed. Wiley*, 1997.

[24] 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, June 1997.

[25] Norbert Vicari and Robert Schedel, "Performance of the GFR-Service with Constant Available Bandwidth," *INFOCOM'99 Proceedings, IEEE*, pp. 567-574 vol.2, 1999.

[26] José Luis González-Sánchez and Jordi Domingo-Pascual, "TAP: Architecture for Trusted Transfers in ATM Networks with Active Switches," *IEEE ATM'2000, Proceedings*, pp. 105-112 2000.

[27] Alex L. G. Hayzelden and John Bigham (Eds.), "Software Agents for Future Communications Systems," *Springer*, 1999.