

# 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*<sup>1</sup> 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 autenticación 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:

---

<sup>1</sup> 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.

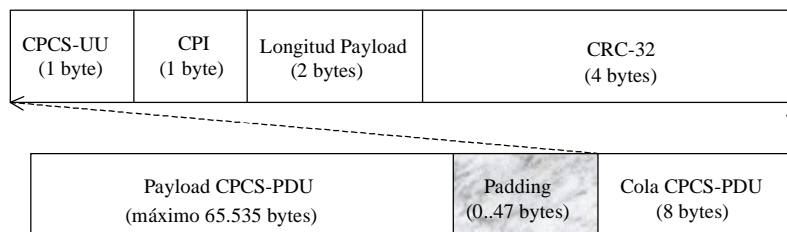
En nuestro caso particular vamos a detenernos en el caso particular de AAL-5 ya que, como adelantamos en el *Capítulo 2*, modificamos y ampliamos este protocolo para conseguir la GoS en la arquitectura TAP.

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 SSSC (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 incurrir en retardos para el nodo receptor, *implosión*<sup>2</sup> 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

<sup>2</sup> La implosión 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)

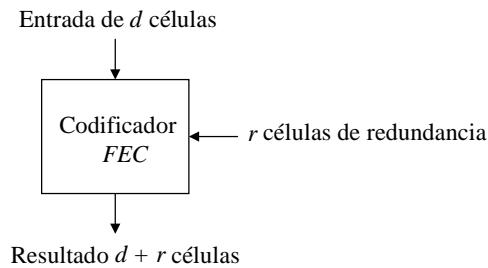
FEC [8-12] es una interesante alternativa a las dos variantes de ARQ pues ofrece fiabilidad sin incrementar la latencia extremo-extremo. Su principio de funcionamiento se basa en la codificación de paquetes en el emisor con información redundante junto con los datos, de forma que sea posible reconstruir 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 añade. 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<sup>3</sup>.

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 él. 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.

<sup>3</sup> 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 decreta la pérdida de paquetes en los árboles y, por tanto, también decreta 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.

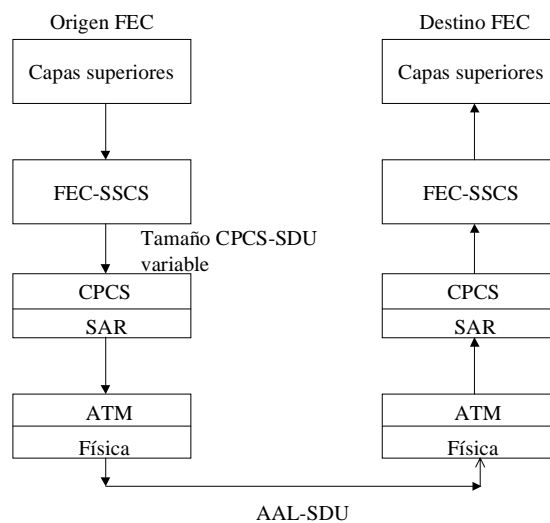
Aunque FEC suele implementarse mediante técnicas hardware usando codificadores y decodificadores, también es implementable mediante software sin pagar demasiado overhead. En el caso de IP, se emplean incluso técnicas de separación de canales apartando por uno de ellos las retransmisiones de datos con 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.

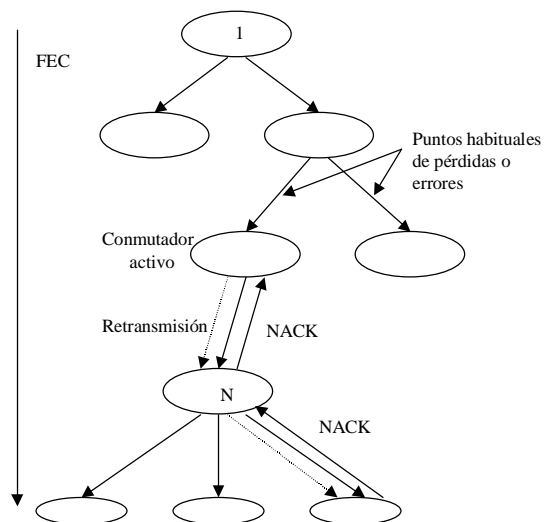
Partiendo de la premisa de la imposibilidad de conseguir la fiabilidad total, hay que destacar que, en comunicaciones punto-a-punto, se ha logrado haciendo que el receptor envíe mensajes ACK positivos al emisor. Para los paquetes de datos recibidos se emplea confirmación positiva y confirmación negativa 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.

Secuencias PDUid (2 bytes)	Longitud Payload (2 bytes)	CRC-32 (4 bytes)
-------------------------------	-------------------------------	---------------------

**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 reenumeración 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 (3<sup>a</sup> 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/srntp>, (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", *31<sup>st</sup> 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 5<sup>th</sup> 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. Kermod "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).